SUPPLEMENTARY MATERIAL OF REINFORCEMENT LEARNING WITH VERIFIABLE YET NOISY REWARDS UNDER IMPERFECT VERIFIERS

Anonymous authors
Paper under double-blind review

LLM USAGE DISCLOSURE

Large Language Models were used solely for English language polishing (grammar and minor wording); all technical contributions—including algorithms, proofs, code, experiments, and analyses—were conceived and validated by the authors.

A MORE EXPERIMENTAL RESULTS

In this section, we also report the average *Pass@8* results on *Qwen2.5-Math-7B* to investigate whether our algorithms can still achieve better performance under more relaxed metrics with upscale models. The experiment setups align that of the main paper.

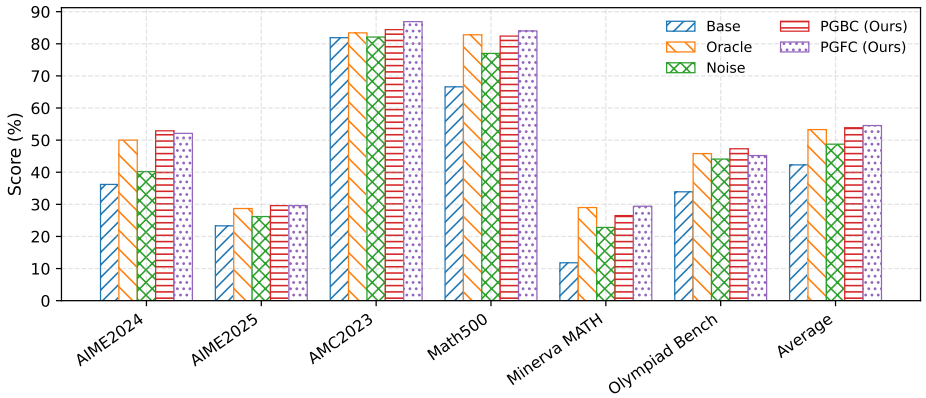

Figure 1: Synthetic-Noise Results on *Qwen2.5-Math-7B*. **Base**: baseline without RL; **Oracle**: Training with clean rewards; **Noise**: Training with noisy verifier rewards; **PGBC**: Training with noise under backward correction; **PGFC**: Training with noise under forward correction.

Table 1: Real-world noise with appeals on negative samples and forward correction on *Qwen2.5-Math-7B*. `Rule`: rule-based rewards; `LV`: direct LLM-judge rewards; `Adds on`: rule-based reward plus LLM appeals on negative samples (no gradient correction); `FCO`: forward correction using online $\hat{\rho}_1$.

Dataset	AIME2024	AIME2025	AMC2023	Math500	Minerva MATH	Olympiad Bench	Average
Base	36.2	23.3	81.9	66.6	11.8	33.9	42.3
Oracle	50.0	28.7	83.4	82.8	**29.0**	45.8	53.3
LV	41.7	22.1	81.9	69.0	13.2	39.4	44.6
Adds on	47.1	**30.4**	**84.4**	80.8	23.5	45.6	52.0
PGFC (Ours)	**54.6**	**30.4**	82.8	**83.2**	**29.0**	**47.6**	**54.6**

As shown in Figure 1 and Table 1, the conclusion in the main paper remains the same: our method can still obtain the best results with both synthetic and real-world noise.

B Implementation Details

We describe how to integrate Algorithm 1 (backward, unbiased reward de-biasing) and Algorithm 2 (forward, gradient-scaled) into *Group Relative Policy Optimization* (GRPO) under both *outcome* and *process* supervision. GRPO samples, for each prompt x, a group of K responses $\{y_i\}_{i=1}^K$ from the behavior policy, computes a *group-normalized advantage* for each sample (or step), and then applies a PPO-style clipped surrogate with a separate KL regularizer to a reference policy; no value network is used. Our modifications are confined to the advantage-construction stage, leaving ratio clipping and KL loss unchanged (details of GRPO in (Shao et al., 2024) and open-source implementations).

Notation (shared). Let π_θ be the current policy and π_{old} the behavior policy. Define token-level ratios $r_{i,t} = \frac{\pi_\theta(y_{i,t}|x,y_{i,<t})}{\pi_{\text{old}}(y_{i,t}|x,y_{i,<t})}$. GRPO's PPO-style surrogate at token t uses an advantage $A_{i,t}$:

$$\mathcal{L}_{\text{grpo}}(\theta) = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\big(r_{i,t}\,A_{i,t},\ \text{clip}(r_{i,t},1\pm\varepsilon)\,A_{i,t}\big)\ -\ \beta\,\text{KL}(\pi_\theta\|\pi_{\text{ref}}),$$

where the KL term is added to the loss rather than folded into the reward. Our corrections only change how $A_{i,t}$ is formed.

Outcome Supervision (one scalar reward per response)

For each i, we observe a binary verifier reward $\tilde{r}_i \in \{0,1\}$.

Algo 1 (Backward) in GRPO. First construct an *unbiased* per-sample reward

$$\widehat{r}_i \ =\ \frac{\tilde{r}_i - \hat{\rho}_0}{1-\hat{\rho}_0-\hat{\rho}_1}.$$

Compute group statistics on $\{\widehat{r}_i\}_{i=1}^K$:

$$\bar{r}\ =\ \frac{1}{K}\sum_{i=1}^{K}\widehat{r}_i, \qquad s\ =\ \sqrt{\frac{1}{K}\sum_{i=1}^{K}(\widehat{r}_i-\bar{r})^2}\,.$$

Define the group-normalized advantage *constant across tokens* of the same response,

$$a_i^{(\text{back})}\ =\ \frac{\widehat{r}_i-\bar{r}}{s+\varepsilon}, \qquad A_{i,t}\ \equiv\ a_i^{(\text{back})},\ \forall t.$$

This is a drop-in replacement for the usual GRPO outcome-advantage, with the sole change being that the group mean/variance are computed over de-noised rewards $\widehat{r}_i$ rather than raw $\tilde{r}_i$.

Algo 2 (Forward) in GRPO. First form the *standard* GRPO outcome-advantage from the *observed* rewards:

$$a_i\ =\ \frac{\tilde{r}_i-\overline{\tilde{r}}}{\text{std}(\{\tilde{r}_j\})+\varepsilon}, \qquad A_{i,t}\ \equiv\ a_i,\ \forall t.$$

Then apply the forward weight determined only by $\hat{\rho}_1$:

$$w_i\ =\ \begin{cases}\hat{\rho}_1-1, & \tilde{r}_i=0,\\ \hat{\rho}_1, & \tilde{r}_i=1,\end{cases} \qquad A_{i,t}\ \leftarrow\ w_i\cdot A_{i,t}\,.$$

Intuitively, the group-normalization is a positive scaling of each sample's (token-shared) factor, so multiplying by w_i implements the same gradient scaling as in REINFORCE, but expressed at the advantage level that GRPO's surrogate consumes. The rest of GRPO (ratio clipping, KL loss) is unchanged.

Process Supervision (step-wise rewards)

Suppose each response y_i has step indices $\text{index}_i(1) < \cdots < \text{index}_i(K_i)$ with step-level observed rewards $\tilde{r}_i^{(j)}$ attached at those indices. GRPO forms step-normalized rewards and turns them into token advantages by backward accumulation over steps.

Algo 1 (Backward) in GRPO-Process. De-noise each step reward:

$$\widehat{r}_i^{(j)} \;=\; \frac{\tilde{r}_i^{(j)} - \hat{\rho}_0}{1 - \hat{\rho}_0 - \hat{\rho}_1}.$$

Normalize across the *group and steps* in the current batch following GRPO's process recipe:

$$\widetilde{r}_i^{(j)} \;=\; \frac{\widehat{r}_i^{(j)} - \mathrm{mean}(\{\widehat{r}_\ell^{(m)}\})}{\mathrm{std}(\{\widehat{r}_\ell^{(m)}\}) + \varepsilon}.$$

Accumulate into token-level advantages (for all tokens t at or before the j-th step boundary):

$$A_{i,t} \;=\; \sum_{\mathrm{index}_i(j) \geq t} \widetilde{r}_i^{(j)}\,.$$

This mirrors the standard GRPO-Process pipeline, but with $\tilde{r}$ replaced by $\widehat{r}$.

Algo 2 (Forward) in GRPO-Process. First follow the *standard* GRPO-Process normalization to obtain $\widetilde{r}_i^{(j)}$ from the *observed* $\tilde{r}_i^{(j)}$, then accumulate the token advantages

$$A_{i,t} \;=\; \sum_{\mathrm{index}_i(j) \geq t} \widetilde{r}_i^{(j)}\,,$$

and finally apply forward weights. If the verifier outputs are binary per step, use

$$w_i^{(j)} \;=\; \begin{cases} \hat{\rho}_1 - 1, & \tilde{r}_i^{(j)} = 0, \\ \hat{\rho}_1, & \tilde{r}_i^{(j)} = 1, \end{cases} \qquad A_{i,t} \;\leftarrow \sum_{\mathrm{index}_i(j) \geq t} \big(w_i^{(j)}\, \widetilde{r}_i^{(j)}\big).$$

If only a *final* binary reward is available, use a single sample-level weight w_i for all steps of y_i (as in outcome supervision) after the standard process accumulation.

Practical notes. (i) **Where to hook.** Implement the corrections exactly at the interface where GRPO converts rewards to (group-)normalized advantages; no change to sampling, clipping, optimizer, or KL regularization. (ii) **Stability.** Backward correction can inflate variance when $1-\hat{\rho}_0-\hat{\rho}_1$ is small; GRPO's group normalization mitigates scale but not variance—use ε and EMA'd statistics as in practice. (iii) **Forward variant.** Because group normalization is a positive rescaling, post-normalization multiplication by w preserves the intended gradient-direction property from the REINFORCE analysis while keeping the rest of GRPO intact. Open-source GRPO implementations follow this decomposition (reward→advantage, then PPO-style surrogate + KL loss).

C Proofs and Derivations

C.1 Proof of Proposition 1

Proof. We compute the expectation of the noisy reward $\tilde{R}$ conditioned on the clean reward R^*, which is a binary variable. By the definition of expectation:

$$\begin{aligned}\mathbb{E}[\tilde{R}] &= 1 \cdot \mathbb{P}(\tilde{R} = 1 \mid R^*) + 0 \cdot \mathbb{P}(\tilde{R} = 0 \mid R^*) \\ &= \mathbb{P}(\tilde{R} = 1 \mid R^*).\end{aligned}$$

We can expand this using the law of total probability, conditioning on the value of $R^* \in \{0, 1\}$:

$$\mathbb{E}[\tilde{R} \mid R^*] = R^* \cdot \mathbb{P}(\tilde{R} = 1 \mid R^* = 1) + (1 - R^*) \cdot \mathbb{P}(\tilde{R} = 1 \mid R^* = 0).$$

From Definition 1, we have $\mathbb{P}(\tilde{R} = 1 \mid R^* = 0) = \rho_0$ and $\mathbb{P}(\tilde{R} = 0 \mid R^* = 1) = \rho_1$, which implies $\mathbb{P}(\tilde{R} = 1 \mid R^* = 1) = 1 - \rho_1$. Substituting these values:

$$\begin{aligned}\mathbb{E}[\tilde{R}] &= R^*(1 - \rho_1) + (1 - R^*)\rho_0 \\ &= R^* - \rho_1 R^* + \rho_0 - \rho_0 R^* \\ &= (1 - \rho_0 - \rho_1)R^* + \rho_0.\end{aligned}$$

This completes the proof. □

C.2 Proof of Theorem 1

Proof. From proposition 1, we have:

$$\mathbb{E}[\tilde{R}] = (1 - \rho_0 - \rho_1)R^* + \rho_0 \,.$$

Taking the full expectation of $\widehat{R}$:

$$\mathbb{E}[\widehat{R}] = \frac{\mathbb{E}[\tilde{R}] - \rho_0}{1 - \rho_0 - \rho_1} = \frac{((1 - \rho_0 - \rho_1)R^* + \rho_0) - \rho_0}{1 - \rho_0 - \rho_1} = R^* \,,$$

showing unbiasedness. □

C.3 Proof of Proposition 2

Proof. The proposition states two claims about the conditional expectation of the forward weights. The weights are defined as:

$$w_{\tilde{R}} \;=\; \begin{cases} w_0 = \rho_1 - 1 & \text{if } \tilde{R} = 0, \\ w_1 = \rho_1 & \text{if } \tilde{R} = 1. \end{cases}$$

The noise model provides the conditional probabilities:

$$\Pr(\tilde{R} = 0 \mid R^* = 1) = \rho_1, \qquad \Pr(\tilde{R} = 1 \mid R^* = 1) = 1 - \rho_1$$
$$\Pr(\tilde{R} = 1 \mid R^* = 0) = \rho_0, \qquad \Pr(\tilde{R} = 0 \mid R^* = 0) = 1 - \rho_0$$

Part 1: Proof of $\mathbb{E}[w_{\tilde{R}} \mid R^* = 1] = 0$ We compute the expectation of $w_{\tilde{R}}$ conditioned on the true reward being positive ($R^* = 1$):

$$\begin{aligned} \mathbb{E}[w_{\tilde{R}} \mid R^* = 1] &= \sum_{k \in \{0,1\}} w_k \cdot \Pr(\tilde{R} = k \mid R^* = 1) \\ &= w_0 \cdot \Pr(\tilde{R} = 0 \mid R^* = 1) + w_1 \cdot \Pr(\tilde{R} = 1 \mid R^* = 1) \\ &= (\rho_1 - 1) \cdot (\rho_1) + (\rho_1) \cdot (1 - \rho_1) \\ &= (\rho_1^2 - \rho_1) + (\rho_1 - \rho_1^2) \\ &= 0. \end{aligned}$$

Part 2: Proof of $\mathbb{E}[w_{\tilde{R}} \mid R^* = 0] = -(1 - \rho_0 - \rho_1)$ Next, we compute the expectation of $w_{\tilde{R}}$ conditioned on the true reward being negative ($R^* = 0$):

$$\begin{aligned} \mathbb{E}[w_{\tilde{R}} \mid R^* = 0] &= \sum_{k \in \{0,1\}} w_k \cdot \Pr(\tilde{R} = k \mid R^* = 0) \\ &= w_0 \cdot \Pr(\tilde{R} = 0 \mid R^* = 0) + w_1 \cdot \Pr(\tilde{R} = 1 \mid R^* = 0) \\ &= (\rho_1 - 1) \cdot (1 - \rho_0) + (\rho_1) \cdot (\rho_0) \\ &= (\rho_1 - \rho_0\rho_1 - 1 + \rho_0) + \rho_0\rho_1 \\ &= \rho_1 + \rho_0 - 1 \\ &= -(1 - \rho_0 - \rho_1). \end{aligned}$$

This proves both claims of the proposition. □

C.4 Proof of Theorem 2

Proof. We want to show that $\mathbb{E}[\Delta\theta] = (1 - \rho_0 - \rho_1)\,\nabla_\theta J(\theta)$, where $\Delta\theta = \frac{1}{M}\sum_{t=1}^{M} h_t$ and $h_t = w_{\tilde{R}} G_t$. By linearity of expectation and assuming i.i.d. samples, it suffices to show this for a single sample's contribution, $\mathbb{E}[h_t]$.

We use the law of total expectation, conditioning on the latent true reward $R^* \in \{0, 1\}$:

$$\begin{aligned} \mathbb{E}[h_t] = \mathbb{E}[w_{\tilde{R}} G_t] &= \mathbb{E}\left[\mathbb{E}[w_{\tilde{R}} G_t \mid R^*]\right] \\ &= \Pr(R^* = 1)\,\mathbb{E}[w_{\tilde{R}} G_t \mid R^* = 1] + \Pr(R^* = 0)\,\mathbb{E}[w_{\tilde{R}} G_t \mid R^* = 0]. \end{aligned}$$

The noise process generating $\tilde{R}$ is independent of the policy's action generation process (which produces G_t), conditional on the true reward R^*. Thus, we can separate the expectations:

$$\mathbb{E}[w_{\tilde{R}} G_t \mid R^*] = \mathbb{E}[w_{\tilde{R}} \mid R^*] \cdot \mathbb{E}[G_t \mid R^*].$$

Using the results from Proposition 2:

- $\mathbb{E}[w_{\tilde{R}} \mid R^* = 1] = 0$.
- $\mathbb{E}[w_{\tilde{R}} \mid R^* = 0] = -(1 - \rho_0 - \rho_1)$.

Substituting these back into the main expectation formula:

$$\begin{aligned}\mathbb{E}[w_{\tilde{R}} G_t] &= \Pr(R^* = 1) \cdot (0) \cdot \mathbb{E}[G_t \mid R^* = 1] + \Pr(R^* = 0) \cdot (-(1 - \rho_0 - \rho_1)) \cdot \mathbb{E}[G_t \mid R^* = 0] \\ &= -(1 - \rho_0 - \rho_1) \cdot \Pr(R^* = 0)\mathbb{E}[G_t \mid R^* = 0] \\ &= -(1 - \rho_0 - \rho_1) \cdot \mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t],\end{aligned}$$

where $\mathbf{1}_{\{\cdot\}}$ is the indicator function. From two fundamental properties of the score function:

1. The unconditional expectation is zero: $\mathbb{E}[G_t] = 0$ (Williams, 1992; Sutton et al., 1999).
2. The clean policy gradient is $\nabla_\theta J(\theta) = \mathbb{E}[R^* G_t]$.

From property 1, we have $\mathbb{E}[G_t] = \mathbb{E}[(\mathbf{1}_{\{R^*=1\}} + \mathbf{1}_{\{R^*=0\}})G_t] = \mathbb{E}[R^* G_t] + \mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t] = 0$. This implies that $\mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t] = -\mathbb{E}[R^* G_t] = -\nabla_\theta J(\theta)$.

Finally, we substitute this back into our expression for the expected update direction:

$$\begin{aligned}\mathbb{E}[h_t] &= \mathbb{E}[w_{\tilde{R}} G_t] \\ &= -(1 - \rho_0 - \rho_1) \cdot \mathbb{E}[\mathbf{1}_{\{R^*=0\}} G_t] \\ &= -(1 - \rho_0 - \rho_1) \cdot (-\nabla_\theta J(\theta)) \\ &= (1 - \rho_0 - \rho_1)\nabla_\theta J(\theta).\end{aligned}$$

Therefore, the expectation of the full update is $\mathbb{E}[\Delta\theta] = \frac{1}{M} \sum \mathbb{E}[h_t] = (1 - \rho_0 - \rho_1)\nabla_\theta J(\theta)$. This completes the proof. □

D Prompt Templates and Training/Evaluation Details

This section records the exact prompt formats and the concrete hyperparameters we used for all experiments in *Reinforcement Learning with Verifiable yet Noisy Rewards under Unreliable Verifiers*. We mirror the level of detail used in recent RLVR appendices and report settings sufficient for full reproducibility from our released code.

D.1 Prompt Templates

Training (generation) prompt. For each math problem x (a plain-text question), the user message is built by concatenating the raw question with a short instruction that elicits chain-of-thought and enforces a verifiable answer format.

```
<user>
{QUESTION}

Let's think step by step and enclose the reasoning process within <
    think> and </think> tags.
The final result in the answer MUST BE within \boxed{}.
</user>
```

During data preprocessing, we write chat-style JSON with a single user turn as shown above and attach the rule-based ground-truth answer for reward checking.

Evaluation (validation/test) prompt. We use the same prompt template as training for validation and test-time generation so that the rule-based verifier can parse the boxed answer consistently.

Verifier I/O. The **rule-based** checker operates on the model's final string and extracts the last `\boxed{...}` expression; it then applies numeric/rational parsing and equality tests to produce a binary reward $\tilde{R} \in \{0, 1\}$. When the **LLM verifier** is enabled, it receives the pair $(\text{problem}, \text{model solution})$ and returns a binary correctness decision used only to estimate the false negative rate ρ_1 over a sliding window. The LLM verifier does not replace the rule-based reward.

D.2 Data Preprocessing

We load the preview split of the math-reasoning corpus and map each example to a chat-style record as above, keeping the reference (ground-truth) answer for programmatic checking.

D.3 Training Configuration

Unless otherwise stated, all runs use GRPO (outcome supervision) with the following constants.

Training (GRPO)	
Train batch size	128
Rollouts per question (group size)	8
Max prompt length (tokens)	512
Max response length (tokens)	3072
Sampling temperature (rollouts)	1.0
Advantage estimator	Group-normalized (GRPO)
KL regularization	Enabled
KL coefficient β	0.001
Entropy coefficient	0.0
Optimizer	AdamW
Learning rate	3e-6
Total epochs	1

Table 2: Core training settings.

Model/backbone. We load the base model from local cache (HuggingFace layout), enable FSDP2 for actor/ref, and use shared-memory weights with remove-padding for efficient vLLM rollouts. The KL is computed w.r.t. a frozen reference initialized from the same base.

D.4 Datasets

Training.

- **DeepScaleR** (PraMamba, 2025): the math-reasoning corpus used for RLVR rollouts and policy updates.

Evaluation. We evaluate with the standard official/problem releases for each benchmark and apply the same rule-based programmatic checking protocol throughout.

- **MATH500** (Lightman et al., 2023): a verifiable subset of math reasoning problems for reliable evaluation.
- **AIME 2024** (HuggingFaceH4, 2024): the 2024 American Invitational Mathematics Examination set.
- **AIME 2025** (OpenCompass, 2025): the 2025 AIME set.
- **AMC 2023** (math-ai, 2025): problems from the 2023 American Mathematics Competitions.
- **Minerva Math** (Lewkowycz et al., 2022): the Minerva math evaluation suite.
- **OlympiadBench** (He et al., 2024): olympiad-level mathematics benchmark.

E False Positive (FP) and False Negative (FN) Examples

We provide FP and FN examples using Qwen2.5-Math-7B base model on MATH 500.

FP


```
{
    "idx": 16,
    "question": "Compute: $1-2+3-4+5- \\dots +99-100$.",
    "model_prediction": "{}",
    "model_response": "Compute: $1+2-3-4+5+6-7-8+ \\dots +97+98-99-100$
    .  \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}.
    \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\
    boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}. \\boxed{{}}.  \\",
    "ground_truth": "-50",
    "rule_based": false,
    "llm_verifier": true,
    "llm_verifier_score": 1.0,
    "llm_verifier_raw_value": "1.0",
    "math_verify": false,
    "math_verify_score": 0.0
  },
```


FN


```
{
    "idx": 92,
    "question": "A pirate searches seven islands for buried treasure.
    If each island has a $\\frac{1}{5}$ chance of having treasure, what
    is the probability that exactly 4 of the islands have treasure?",
    "answer": "\\frac{448}{15625}",
    "model_prediction": "0.028672",
    "model_response": "To solve this problem, we can use the binomial
    probability formula. The probability of having exactly $k$ successes
     in $n$ independent Bernoulli trials (each with success probability
    $p$) is given by:\n\n\\[ P(X = k) = \\binom{n}{k} p^k (1-p)^{n-k}
    \\]\n\nHere, $n = 7$ (the number of islands), $k = 4$ (the number of
     islands with treasure), and $p = \\frac{1}{5}$ (the probability
    that a single island has treasure).\n\nThe binomial coefficient $\\
    binom{n}{k}$ is the number of ways to choose $k$ successes out of
    $n$ trials, and it is calculated as:\n\n\\[ \\binom{n}{k} = \\frac{n
    !}{k!(n-k)!} \\]\n\nSo, the probability that exactly 4 of the 7
    islands have treasure is:\n\n\\[ P(X = 4) = \\binom{7}{4} \\left(\\
    frac{1}{5}\\right)^4 \\left(\\frac{4}{5}\\right)^{3} \\]\n\nLet's
    calculate this using Python.\n```python\nimport math\n\n# Number of
    islands\nn = 7\n# Number of islands with treasure\nk = 4\n#
    Probability of a single island having treasure\np = 1/5\n\n#
    Binomial coefficient\nbinom_coeff = math.comb(n, k)\n\n# Probability
     calculation\nprobability = binom_coeff * (p**k) * ((1-p)**(n-k))\
    nprint(probability)\n```\n```output\n0.028672000000000001\n```\nThe
    probability that exactly 4 of the 7 islands have treasure is \\(\\
    boxed{0.028672}\\).",
    "ground_truth": "\\frac{448}{15625}",
    "rule_based": false,
```

```
    "llm_verifier": false,
    "llm_verifier_score": 0.0,
    "llm_verifier_raw_value": "0.0",
    "math_verify": true,
    "math_verify_score": 1.0
  }
```


REFERENCES

Chaoqun He, Renjie Luo, Yuzhuo Bai, et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. URL https://aclanthology.org/2024.acl-long.211.pdf.

HuggingFaceH4. Aime 2024 (dataset card). Hugging Face, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/aime_2024.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/18abbeef8cfe9203fdf9053c9c4fe191-Paper-Conference.pdf.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

math-ai. Amc 2023 (dataset card). Hugging Face, 2025. URL https://huggingface.co/datasets/math-ai/amc23.

OpenCompass. Aime 2025 (dataset card). Hugging Face, 2025. URL https://huggingface.co/datasets/opencompass/AIME2025.

PraMamba. Deepscaler. GitHub repository, 2025. URL https://github.com/PraMamba/DeepScaleR.

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. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.