DeepSeekMath

Pushing the Limits of Mathematical Reasoning in Open Language Models

Natan da Silveira Ferreira, Nicolas da Silva Wischermann

12 de dezembro de 2025

Motivação

- Large Language Models (LLM's) no domínio da Matemática têm sido importantes ferramentas para auxiliar humanos na resolução de problemas matemáticos complexos

- **Problema:** Modelos de ponta não são acessíveis publicamente
  - GPT-4, Gemini-Ultra são fechados (closed source).
  - Modelos open-source atuais ficam muito atrás no desempenho dos citados acima.

- **Solução proposta:** DeepSeekMath
  - Modelo de linguagem de domínio específico para Matemática
  - Supera capacidades matemáticas dos modelos open-source
  - Aproxima-se do desempenho do GPT-4 em benchmarks acadêmicos

Principais Contribuições

- Evidências do Common Crawl como dataset valioso para fins de modelos matemáticos.
- Número de parâmetros não representa qualidade do modelo.
- Modelos previamente treinados para resolver códigos de programação apresentam melhor resposta ao treinamento matemático.
- GRPO como alternativa ao PPO em Aprendizado por Reforço.

Visão Geral

A fim de alcançar tais objetivos, são necessários diferencias como:

- Grande corpus de pré-treinamento matemático (120B tokens)
- Inicialização a partir de modelo de código (DeepSeek-Coder)
- Algoritmo GRPO para otimização por reforço eficiente

Dataset

- Dataset inicial para o treinamento do modelo: DeepSeekMathSeed - pequeno, mas com dados de alta qualidade recuperado do dataset OpenWebMath.
- Recupera-se páginas similares as existentes no DeepSeekMathSeed do Common Crawl (exemplos positivos e negativos)
- Popula-se o DeepSeekMathCorpus
- Observação: As páginas que contém questões ou resoluções de problemas de Benchmarks são descartadas.

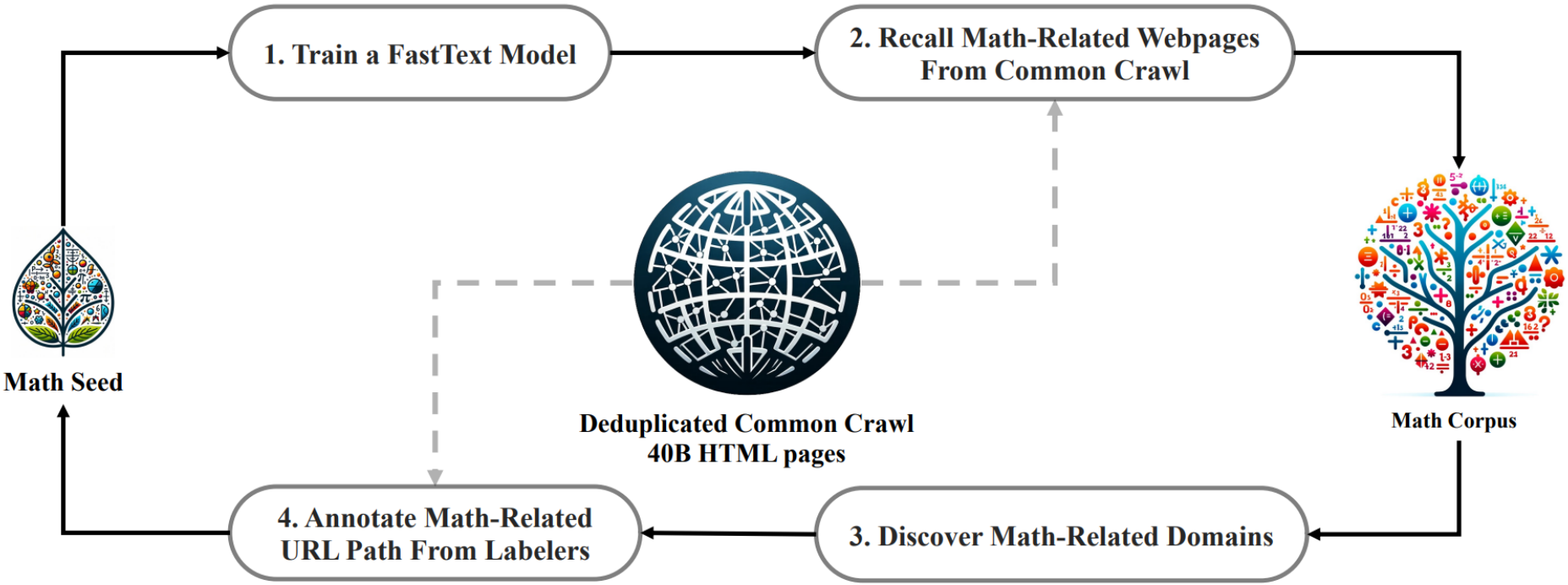


Figure 2 | An iterative pipeline that collects mathematical web pages from Common Crawl.

Avaliando a qualidade dos dados do DeepSeekMath-Corpus

Math Corpus	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
No Math Training	N/A	2.9%	3.0%	2.9%	15.6%	19.5%	12.3%	0.8%	17.9%
MathPile	8.9B	2.7%	3.3%	2.2%	12.5%	15.7%	1.2%	0.0%	2.8%
OpenWebMath	13.6B	11.5%	8.9%	3.7%	31.3%	29.6%	16.8%	0.0%	14.2%
Proof-Pile-2	51.9B	14.3%	11.2%	3.7%	43.8%	29.2%	19.9%	5.1%	11.7%
DeepSeekMath Corpus	**120.2B**	**23.8%**	**13.6%**	**4.8%**	**56.3%**	**33.1%**	**41.5%**	**5.9%**	**23.6%**

Table 1 | Performance of DeepSeek-LLM 1.3B trained on different mathematical corpora, evaluated using few-shot chain-of-thought prompting. Corpus sizes are calculated using our tokenizer with a vocabulary size of 100K.

Métricas de Avaliação

- Benchmarks de problemas matemáticos em Inglês e em Chinês, do nível fundamental ao universitário.
- Demonstrações de Teoremas.
- Compreensão de Linguagem Natural, Raciocínio e Código.

DeepSeekMath-Base 7B

- **Modelo base** com fortes habilidades de raciocínio, especialmente em matemática: DeepSeek-Coder-Base-v1.5 7B
- **Treinamento:** 500B tokens
- **Distribuição dos dados:**
  - 56% DeepSeekMath Corpus
  - 4% AlgebraicStack
  - 10% arXiv
  - 20% Código GitHub
  - 10% Linguagem natural (Inglês e Chinês)
- **Avaliação abrangente:**
  - Soluções matemáticas autônomas (sem ferramentas externas)
  - Resolução com uso de ferramentas
  - Prova formal de teoremas
  - Compreensão de linguagem natural
  - Raciocínio geral
  - Habilidades de programação

Resultados em resoluções de problemas passo a passo

Model	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
Closed-Source Base Model									
Minerva	7B	16.2%	14.1%	7.7%	-	35.6%	-	-	-
Minerva	62B	52.4%	27.6%	12.0%	-	53.9%	-	-	-
Minerva	540B	58.8%	33.6%	17.6%	-	63.9%	-	-	-
Open-Source Base Model									
Mistral	7B	40.3%	14.3%	9.2%	71.9%	51.1%	44.9%	5.1%	23.4%
Llemma	7B	37.4%	18.1%	6.3%	59.4%	43.1%	43.4%	11.9%	23.6%
Llemma	34B	54.0%	25.3%	10.3%	71.9%	52.9%	56.1%	11.9%	26.2%
DeepSeekMath-Base	7B	**64.2%**	**36.2%**	**15.4%**	**84.4%**	**56.5%**	**71.7%**	**20.3%**	**35.3%**

Table 2 | Comparisons between DeepSeekMath-Base 7B and strong base models on English and Chinese mathematical benchmarks. Models are evaluated with chain-of-thought prompting. Minerva results are quoted from Lewkowycz et al. (2022a).

Resultados destacados (Tabela 2):

- Liderança em todos os 8 benchmarks entre modelos base open-source
- Supera modelos como Mistral 7B e Llemma 34B
- **MATH (nível competição):** +10% absoluto sobre outros open-source
- **Feito notável:** Supera Minerva 540B (modelo fechado 77x maior)

DeepSeekMath-Instruct 7B: Resultado do ajuste fino do DeepSeekMath-Base 7B

Model	Size	English Benchmarks		Chinese Benchmarks	
		GSM8K	MATH	MGSM-zh	CMATH
Chain-of-Thought Reasoning					
Closed-Source Model					
Gemini Ultra	-	94.4%	53.2%	-	-
GPT-4	-	92.0%	52.9%	-	86.0%
Inflection-2	-	81.4%	34.8%	-	-
GPT-3.5	-	80.8%	34.1%	-	73.8%
Gemini Pro	-	86.5%	32.6%	-	-
Grok-1	-	62.9%	23.9%	-	-
Baichuan-3	-	88.2%	49.2%	-	-
GLM-4	-	87.6%	47.9%	-	-
Open-Source Model					
InternLM2-Math	20B	82.6%	37.7%	-	-
Qwen	72B	78.9%	35.2%	-	-
Math-Shepherd-Mistral	7B	84.1%	33.0%	-	-
WizardMath-v1.1	7B	83.2%	33.0%	-	-
DeepSeek-LLM-Chat	67B	84.1%	32.6%	74.0%	80.3%
MetaMath	70B	82.3%	26.6%	66.4%	70.9%
SeaLLM-v2	7B	78.2%	27.5%	64.8%	-
ChatGLM3	6B	72.3%	25.7%	-	-
WizardMath-v1.0	70B	81.6%	22.7%	64.8%	65.4%
DeepSeekMath-Instruct	7B	82.9%	46.8%	73.2%	84.6%
DeepSeekMath-RL	7B	**88.2%**	**51.7%**	**79.6%**	**88.8%**
Tool-Integrated Reasoning					
Closed-Source Model					
GPT-4 Code Interpreter	-	97.0%	69.7%	-	-
Open-Source Model					
InternLM2-Math	20B	80.7%	54.3%	-	-
DeepSeek-LLM-Chat	67B	86.7%	51.1%	76.4%	85.4%
ToRA	34B	80.7%	50.8%	41.2%	53.4%
MAmmoTH	70B	76.9%	41.8%	-	-
DeepSeekMath-Instruct	7B	83.7%	57.4%	72.0%	84.3%
DeepSeekMath-RL	7B	**86.7%**	**58.8%**	**78.4%**	**87.6%**

Evolução do GRPO nas Publicações DeepSeek

- DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models
- DeepSeek-V2: A Strong, Economical, and Efficient MoE LLM
- DeepSeek-Coder-V2: Breaking the Barrier of Closed-Source Models in Code Intelligence
- DeepSeek-Prover-V1.5: Harnessing Proof Assistant Feedback for Reinforcement Learning and Monte-Carlo Tree Search
- DeepSeek-V3 Technical Report
- DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via RL

GRPO

- O GRPO é um algoritmo de Reinforcement Learning usado para treinar modelos de linguagem
- Ele é muito parecido com o PPO, e foi inspirado nele.
- Temos um problema: não faz sentido explicar o GRPO sem entender o PPO primeiro, e ambos são algoritmos muito complicados:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}$$

- Vamos, então, tentar entender essa fórmula monstruosa!

Formulação do Problema: RL para Matemática

- Considere um dataset $\mathcal{D}$ de problemas (prompts).
- **Ações:** Para cada problema, a LLM gera uma resposta (sequência de tokens) y, que pode conter cadeias de pensamento, cálculos, etc. Podemos instruir via prompt para que a resposta final esteja em um formato específico (ex: `<SOLUTION>s</SOLUTION>`).
- **Recompensas:** A Recompensa $R(x, y)$ pode ser uma função binária simples (1 se correto, 0 se errado) ou um modelo de recompensa complexo treinado para avaliar o processo.
- **Objetivo:** Maximizar a recompensa esperada sob a política π_θ:

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\, y\sim\pi_\theta(\cdot|x)}[R(x, y)]$$

- **Treinamento:** Treinamos a LLM com gradiente ascendente para maximizar o objetivo.
- **Obs:** Às vezes omitiremos x e $\mathcal{D}$ para simplificar a notação.

Gradiente Ascendente

Para maximizar $J(\theta)$, precisamos estimar seu gradiente. Usando a identidade $\nabla \log \pi = \frac{\nabla \pi}{\pi}$:

$$\nabla_\theta J(\theta) = \nabla_\theta \left(\mathbb{E}_{y \sim \pi_\theta} [R(y)] \right) = \mathbb{E}_{y \sim \pi_\theta} \left[R(y) \nabla_\theta \log \pi_\theta(y) \right]$$

Note que a probabilidade da resposta y é o produto das probabilidades dos tokens:

$$\pi_\theta(y) = \prod_{t=1}^{|O|} \pi_\theta(o_t | o_{<t})$$

Processo de Treinamento e Minibatches

Como treinamos na prática para eficiência computacional?

1. **Coleta:** Usamos uma política fixa $\pi_{\theta_{old}}$ para gerar respostas para vários problemas em paralelo.
2. **Dataset Temporário:** Formamos um batch $\mathcal{B}$ de dados coletados $(x, y) \in \mathcal{B}$.
3. **Updates:** Dividimos esse batch em minibatches B_i e realizamos múltiplas atualizações sequenciais nos parâmetros θ.

Estimador do Gradiente no Minibatch:

$$\nabla_\theta J(\theta) \approx \frac{1}{|B_i|} \sum_{(x,y)\in B_i} R(x, y) \nabla_\theta \log \pi_\theta(y|x)$$

Isso nos leva a um problema de viés quando fazemos mais de um update com os dados gerados pela $\pi_{\theta_{old}}$.

O Problema do Viés nos Updates Sequenciais

Suponha que realizamos updates sequenciais $\theta_{old} \to \theta' \to \theta'' \ldots$ usando os dados coletados inicialmente com $\pi_{\theta_{old}}$.

- **Passo 1 ($\theta = \theta_{old}$):** Queremos $\nabla J(\theta_{old})$.

$$\nabla J(\theta_{old}) = \mathbb{E}_{y \sim \pi_{\theta_{old}}}[R(y) \nabla \log \pi_{\theta_{old}}(y)]$$

  Estimador: $\frac{1}{N} \sum R(y_i) \nabla \log \pi_{\theta_{old}}(y_i)$ com amostras $y_i \sim \pi_{\theta_{old}}$.
  (Correto: amostras vêm da distribuição alvo).

- **Passo 2 ($\theta = \theta'$):** Queremos $\nabla J(\theta')$.

$$\nabla J(\theta') = \mathbb{E}_{y \sim \pi_{\theta'}}[R(y) \nabla \log \pi_{\theta'}(y)]$$

  Estimador: $\frac{1}{N} \sum R(y_i) \nabla \log \pi_{\theta'}(y_i)$ com amostras $y_i \sim \pi_{\theta_{old}}$.
  (Incorreto: estimamos gradiente de $\pi_{\theta'}$ usando amostras de $\pi_{\theta_{old}}$).

Isso introduz um viés, pois estamos otimizando uma distribuição usando amostras de outra. Além disso, a variância desse método é muito alta. Como podemos resolver isso?

Importance Sampling e Vantagem

Para corrigir o viés e reduzir variância, derivamos um novo objetivo. Sabemos que $\nabla_\theta \sum_y \pi_\theta(y) b = b \nabla_\theta 1 = 0$. Logo, podemos subtrair um baseline b sem alterar o gradiente.

$$\nabla_\theta J(\theta) = \nabla_\theta \sum_y \pi_\theta(y) R(y) = \nabla_\theta \sum_y \pi_\theta(y)(R(y) - b)$$

$$= \nabla_\theta \sum_y \pi_{\theta_{old}}(y) \frac{\pi_\theta(y)}{\pi_{\theta_{old}}(y)} (R(y) - b)$$

$$= \nabla_\theta \mathbb{E}_{y \sim \pi_{\theta_{old}}} \left[\frac{\pi_\theta(y)}{\pi_{\theta_{old}}(y)} (R(y) - b) \right]$$

Agora a esperança é sobre $\pi_{\theta_{old}}$, a distribuição dos dados coletados.
Definimos a **Vantagem** $A(y) = R(y) - b$.

Objetivo Substituto (J_{IS}):

$$J_{IS}(\theta) = \mathbb{E}_{y \sim \pi_{\theta_{old}}} \left[\frac{\pi_\theta(y)}{\pi_{\theta_{old}}(y)} A(y) \right] = \mathbb{E}_{y \sim \pi_{\theta_{old}}} \left[\prod_t \frac{\pi_\theta(o_t | o_{<t})}{\pi_{\theta_{old}}(o_t | o_{<t})} A(y) \right]$$

Objetivo Token-wise: Soma vs. Produto

O objetivo derivado (J_{IS}) usa o produto das razões de probabilidade (por token), o que gera alta variância. Para diminuir a variância, usamos uma aproximação $L(\theta)$:

$$L(\theta) = \mathbb{E}_{y \sim \pi_{\theta_{old}}} \left[\sum_t \frac{\pi_\theta(o_t|o_{<t})}{\pi_{\theta_{old}}(o_t|o_{<t})} A(y) \right]$$

Comparação dos Gradientes:

- $\nabla J_{IS} = \mathbb{E}\left[\frac{\pi_\theta(y)}{\pi_{\theta_{old}}(y)} A \nabla \log \pi_\theta(y)\right] = \mathbb{E}\left[\sum_t \frac{\pi_\theta(y)}{\pi_{\theta_{old}}(y)} A \nabla \log \pi_\theta(o_t|o_{<t})\right]$
- $\nabla L \quad = \mathbb{E}\left[\sum_t \frac{\pi_\theta(o_t|o_{<t})}{\pi_{\theta_{old}}(o_t|o_{<t})} A \nabla \log \pi_\theta(o_t|o_{<t})\right]$

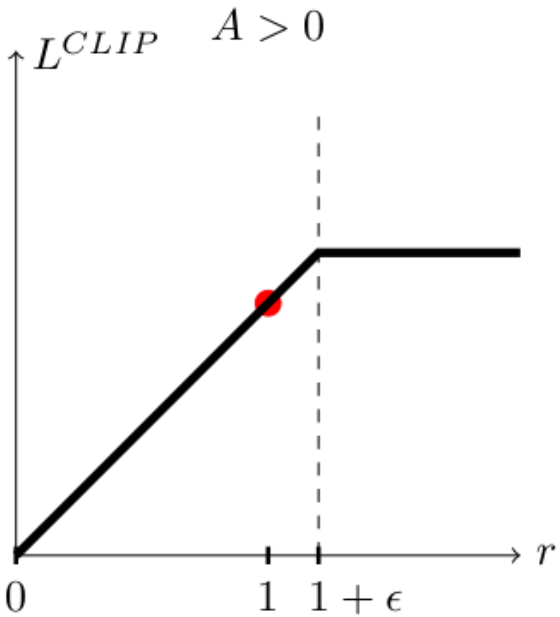
Em $\theta = \theta_{old}$, ambos coincidem, mas ∇L tem variância muito menor. No entanto, costumamos dividir o somatório pelo tamanho da sequência:

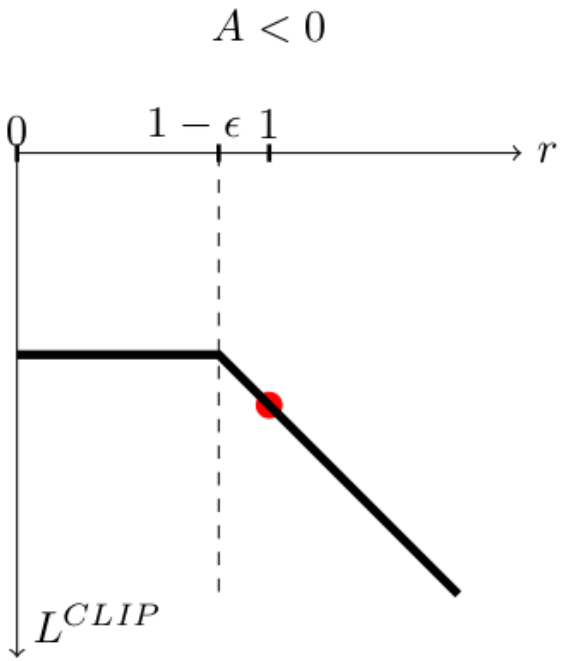
$$L(\theta) = \mathbb{E}_{y \sim \pi_{\theta_{old}}} \left[\frac{1}{|O|} \sum_t \frac{\pi_\theta(o_t|o_{<t})}{\pi_{\theta_{old}}(o_t|o_{<t})} A(y) \right]$$

Clipping: Limitando o Colapso da Política

- Se a nova política mudar drasticamente, a razão $r_t = \frac{\pi_\theta(o_t|o_{<t})}{\pi_{\theta_{old}}(o_t|o_{<t})}$ pode ser muito grande ou muito pequena e levar a *Policy Collapse*.
- Para resolver isso, paramos de atualizar π_θ quando ou o token é bom e r_t cresceu acima de $1+\epsilon$ ou o token é ruim e r_t caiu mais do que $1-\epsilon$:

$$L^{Clip} = \mathbb{E}\left[\frac{1}{|O|}\sum_t \min(r_t A, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)A)\right]$$





Chegando no PPO e GRPO

Agora estamos chegando perto do PPO e do GRPO. A fórmula que temos para o objetivo agora é:

$$L^{Clip} = \mathbb{E}\left[\frac{1}{|O|}\sum_{t=1}^{|O|}\min\left[\frac{\pi_\theta(o_t|o_{<t})}{\pi_{\theta_{old}}(o_t|o_{<t})}A, \text{clip}\left(\frac{\pi_\theta(o_t|o_{<t})}{\pi_{\theta_{old}}(o_t|o_{<t})}, 1-\epsilon, 1+\epsilon\right)A\right]\right]$$

As fórmulas dos objetivos do PPO e GRPO são:

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)]\frac{1}{|o|}\sum_{t=1}^{|o|}\min\left[\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}, 1-\varepsilon, 1+\varepsilon\right)A_t\right]$$

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}$$

Vamos agora ver as duas coisas que precisamos adicionar para chegar nessas fórmulas: uma fórmula explícita para a vantagem A_t e a divergência KL.

PPO vs GRPO: Cálculo da Vantagem A_t

A principal diferença entre os dois algoritmos está em como eles estimam o baseline b para $A_t = R_t - b$.

PPO:

- Usa uma **Value Network** $V_\phi(x)$ (Critic) como baseline para calcular a vantagem do token t:

$$A_t^{PPO} = R_t - V_\phi(o_{<t})$$

- **Custo:** Carregar modelo extra (Critic) do tamanho da LLM.

GRPO:

- Gera G saídas $\{y_1 \dots y_G\}$ para o mesmo prompt, cada uma com um reward, $\{R_1, ..., R_G\}$. Assim, usamos a média e desvio padrão das recompensas para calcular a vantagem do token t:

$$A_i^{GRPO} = \frac{R_i - \mu_{grupo}}{\sigma_{grupo}}$$

- **Vantagem:** Elimina o Critic.
- Note que a vantagem não depende do token, só do resultado inteiro, então ela é igual para todos os tokens de um output.

PPO vs GRPO: Objetivos Finais e Divergência KL

Para garantir que o modelo em treinamento não se distancie muito do modelo base, regularizamos o modelo calculando a divergência KL entre ele (π_θ) e o modelo base (π_{ref}).

PPO:

Adiciona a penalidade na recompensa, $R_t \leftarrow R_t - \beta \log \frac{\pi}{\pi_{ref}}$:

$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min \left[\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}, 1-\varepsilon, 1+\varepsilon \right) A_t \right]$$

GRPO:

O termo KL é subtraído explicitamente da função objetivo:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left\{ \min \left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})} \hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon \right) \hat{A}_{i,t} \right] - \beta \mathbb{D}_{KL}\left[\pi_\theta || \pi_{ref}\right] \right\}$$

Arquitetura: Removendo o Critic

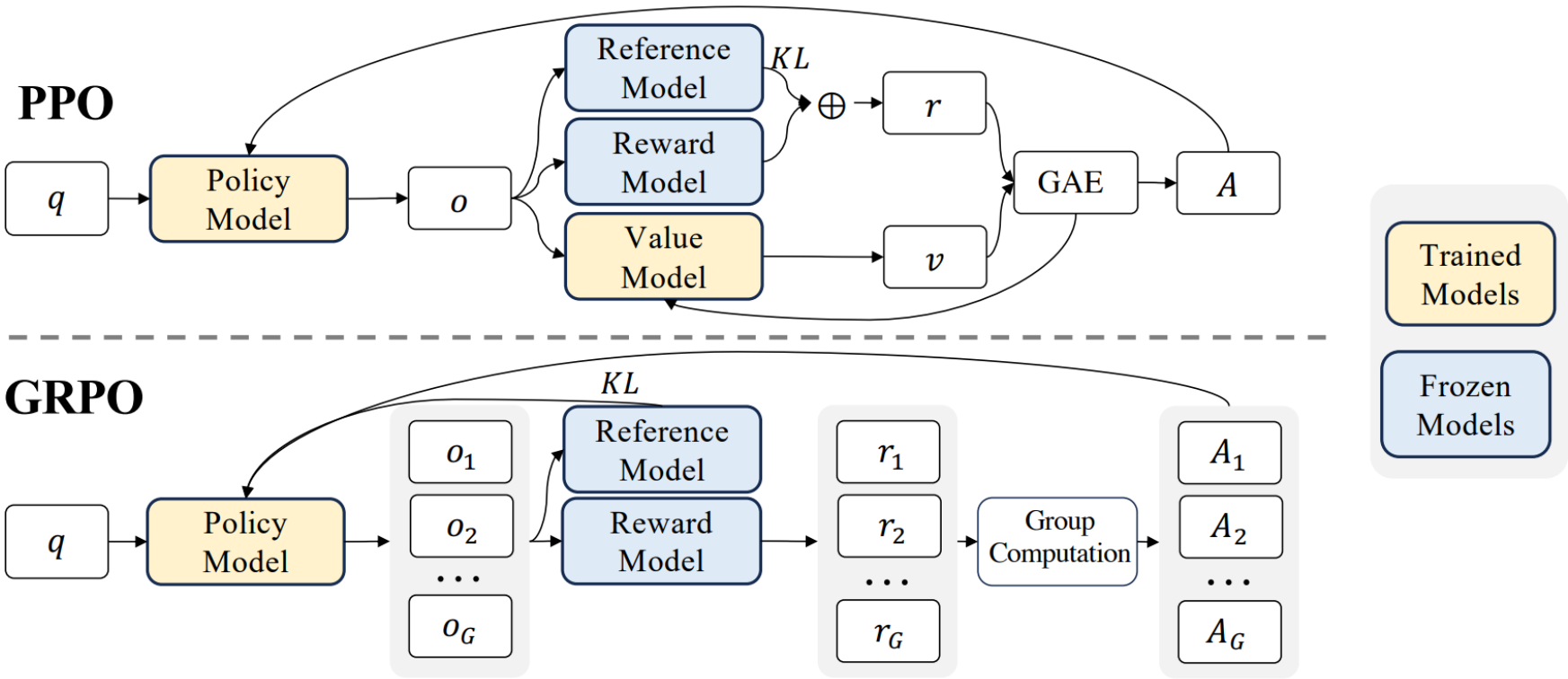
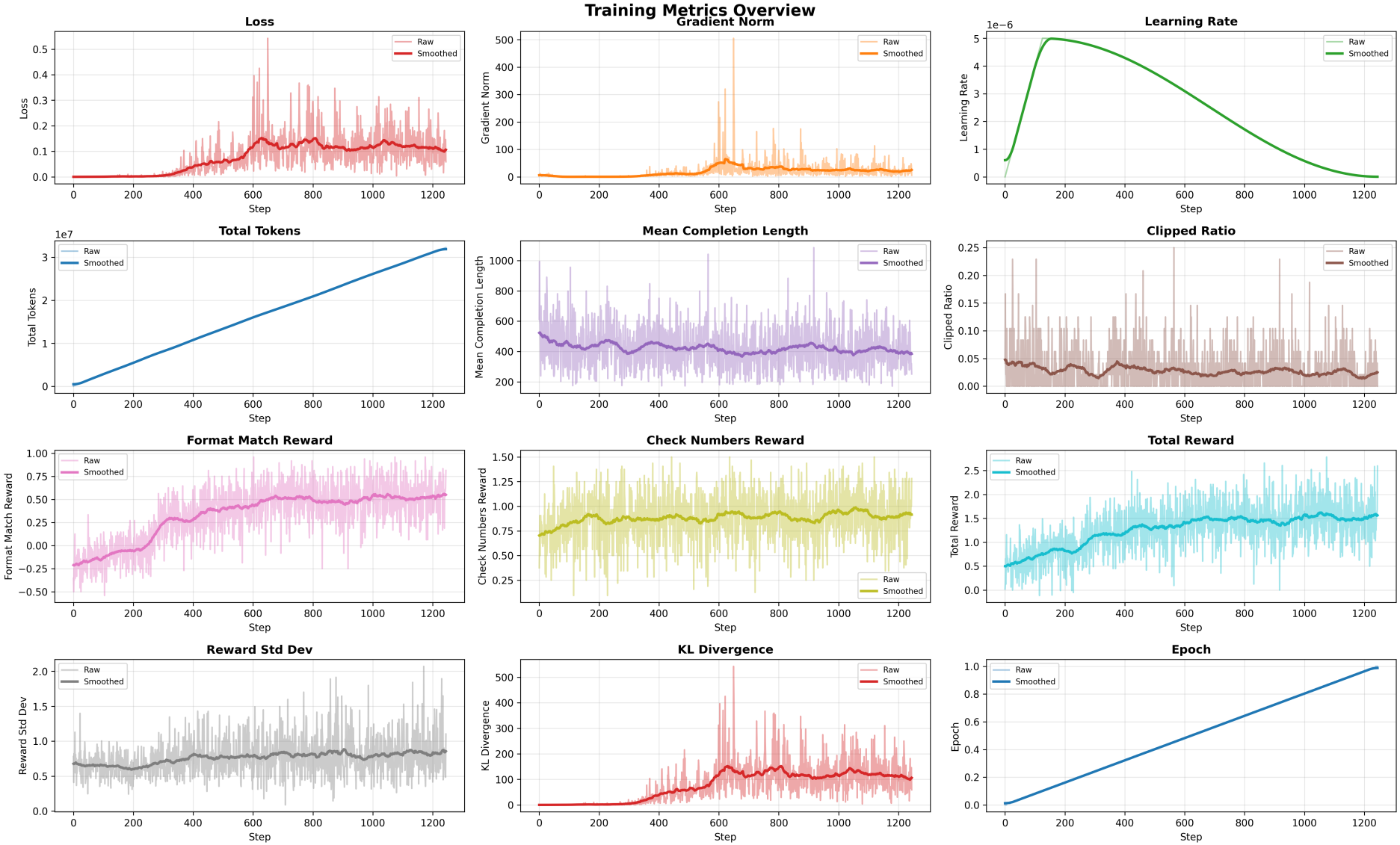


Figure 4 | Demonstration of PPO and our GRPO. GRPO foregoes the value model, instead estimating the baseline from group scores, significantly reducing training resources.

Setup Experimental para o GRPO

- **Dataset**: GSM8K, dataset de problemas simples de matemática (train split com $7,473$ problemas)
- **Modelo**: Gemma-3 1B Instruct com LoRA fine-tuning (rank=8, max seq length=2048)
- **GPU e tempo**: NVIDIA RTX 4090, 1 época completa em ~ 41 horas, batch size 48.
- **Bibliotecas**: Unsloth (LoRA), TRL (GRPO)
- **Reward functions**: Recompensa para formatação e respostas corretas.

Resultados do Treinamento



Resultados no GSM8K