Towards Execution-Grounded Automated AI Research

AI4Research Workshop @ AAAI 2026

Chenglei Si
clsi@stanford.edu

Scientific research is important, but bounded by human expertise.

nature

Explore content · About the journal · Publish with us

nature › articles › article

Article | Published: 04 January 2023

Papers and patents are becoming less disruptive over time

Michael Park, Erin Leahey & Russell J. Funk

Nature **613**, 138–144 (2023) | Cite this article

380k Accesses | **275** Citations | **4577** Altmetric | Metrics

Are Ideas Getting Harder to Find?†

By Nicholas Bloom, Charles I. Jones, John Van Reenen, and Michael Webb*

Long-run growth in many models is the product of two terms: the effective number of researchers and their research productivity. We present evidence from various industries, products, and firms showing that research effort is rising substantially while research productivity is declining sharply. A good example is Moore's Law. The number of researchers required today to achieve the famous doubling of computer chip density is more than 18 times larger than the number required in the early 1970s. More generally, everywhere we look we find that ideas, and the exponential growth they imply, are getting harder to find. (*JEL* D24, E23, O31, O47)

Meanwhile, LLMs are getting better.

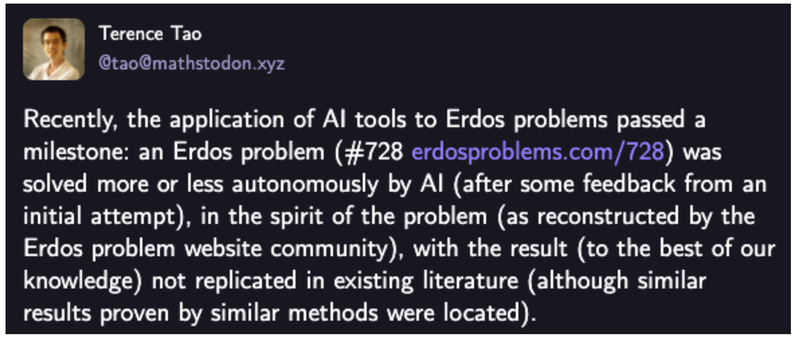
July 21, 2025 Research

Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the International Mathematical Olympiad

OpenAI ✓
@OpenAI

We've scored highly enough to achieve gold at this year's IOI online competition with a reasoning system — placing #6 when ranked with humans and #1 when ranked with other AIs.

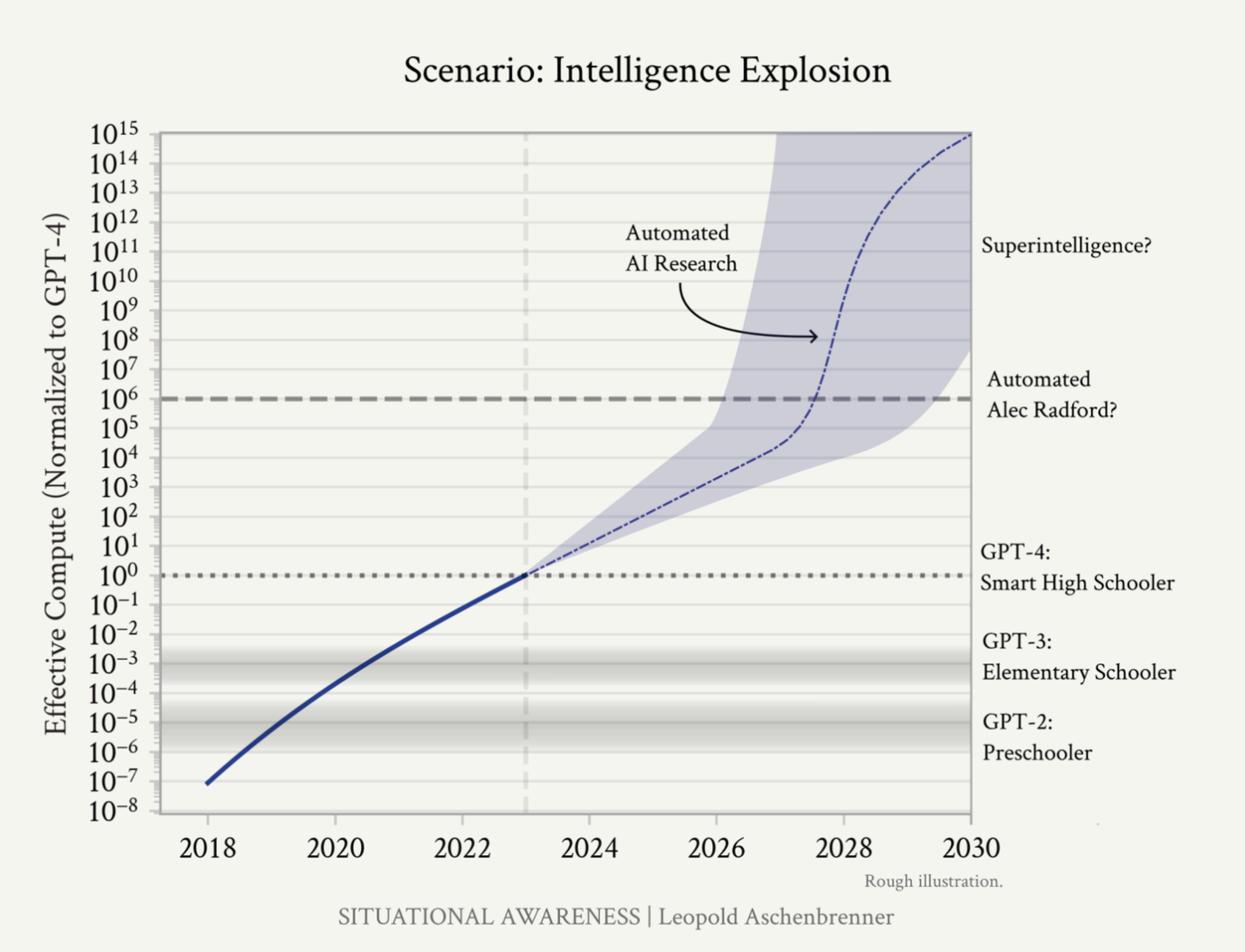
In just a few weeks:
• 2nd at AtCoder
• Gold medal-level at IMO
• Gold medal-level at IOI



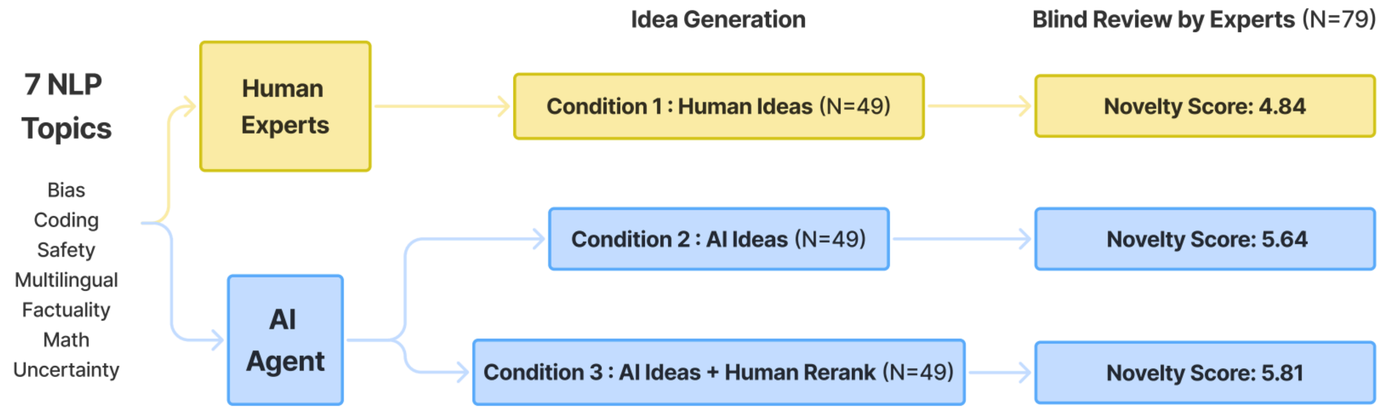
Terence Tao
@tao@mathstodon.xyz

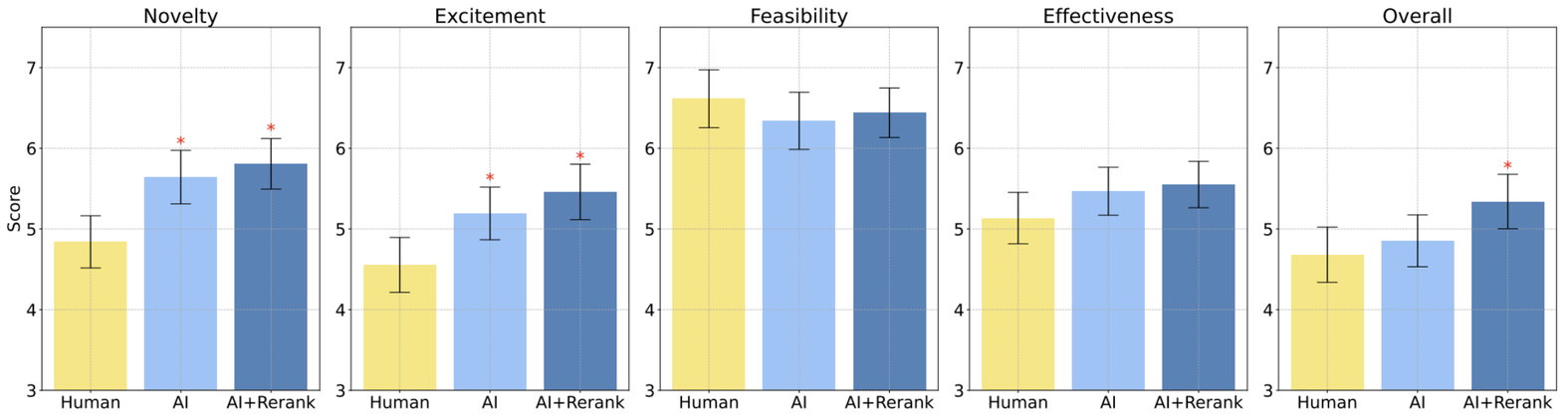
Recently, the application of AI tools to Erdos problems passed a milestone: an Erdos problem (#728 erdosproblems.com/728) was solved more or less autonomously by AI (after some feedback from an initial attempt), in the spirit of the problem (as reconstructed by the Erdos problem website community), with the result (to the best of our knowledge) not replicated in existing literature (although similar results proven by similar methods were located).

Vision: Automated AI Research



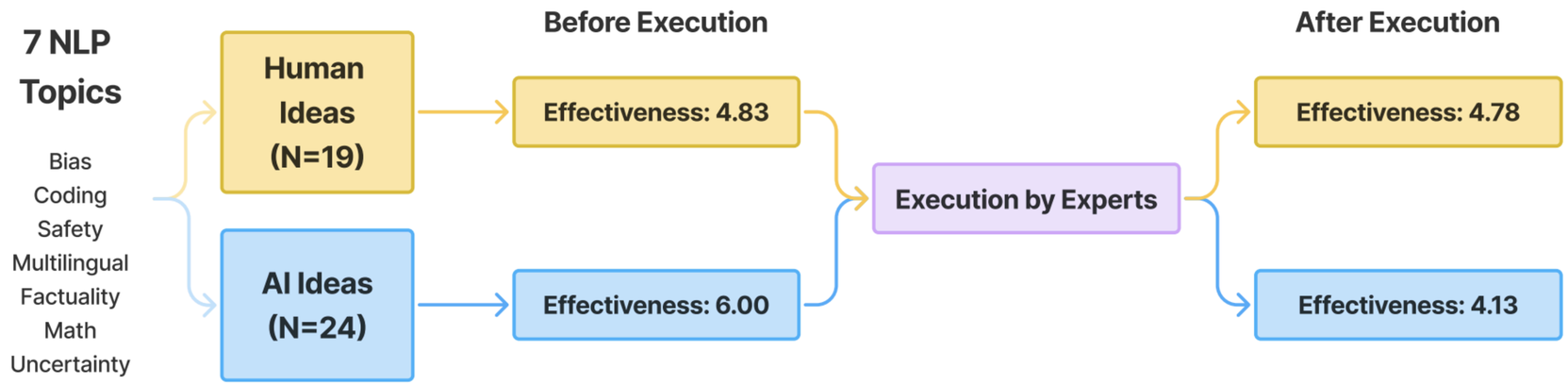
Prerequisite: Generate Good Ideas!

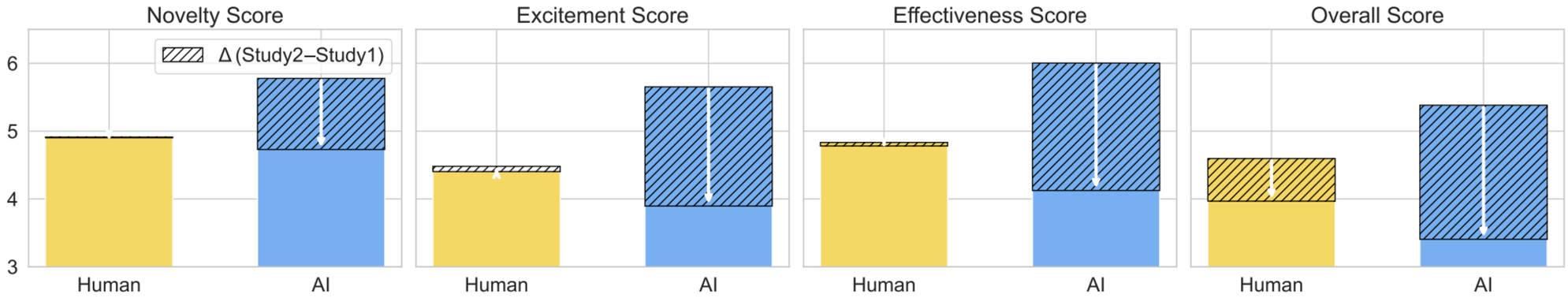




Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers, ICLR 2025

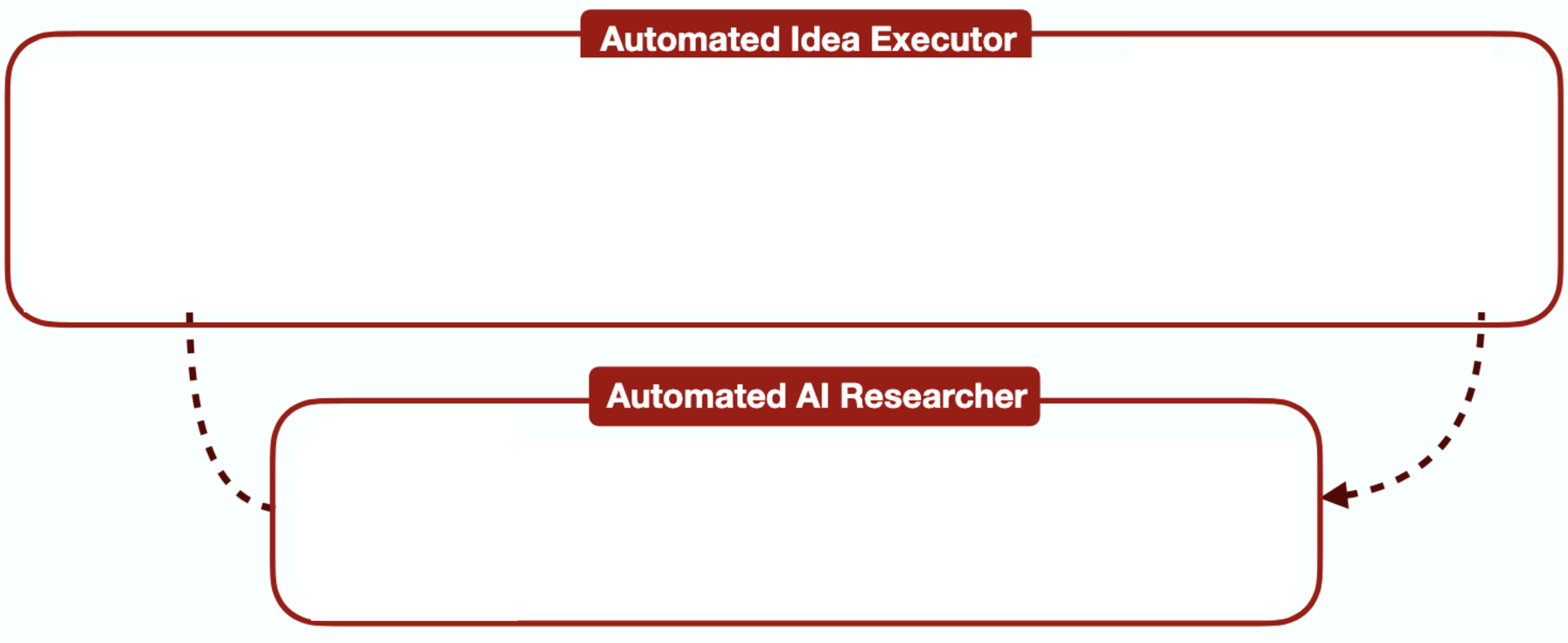
Prerequisite: Generate Good Ideas!



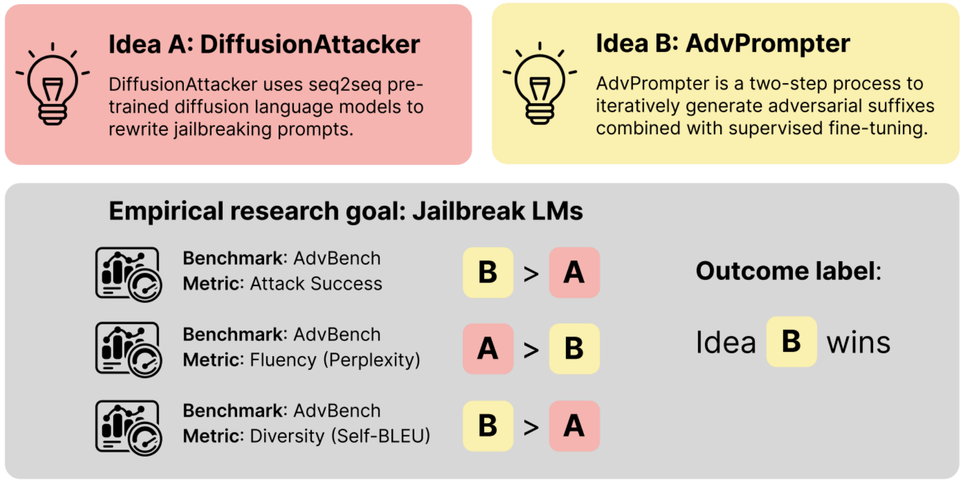


The Ideation-Execution Gap: Execution Outcomes of LLM-Generated versus Human Research Ideas, Under Review

“Obvious Solution”: Execution Grounding



You Might Ask: Can we approximate the execution reward?



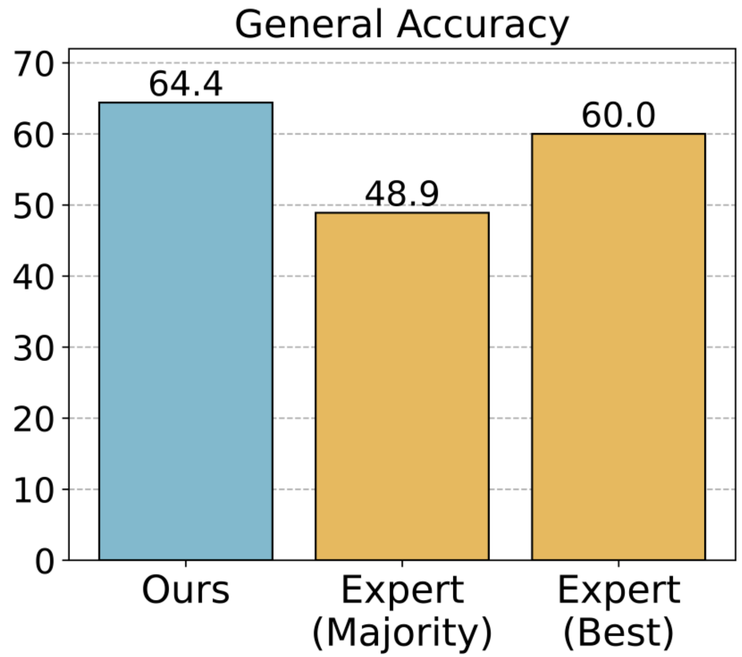


Table 1: Data source.

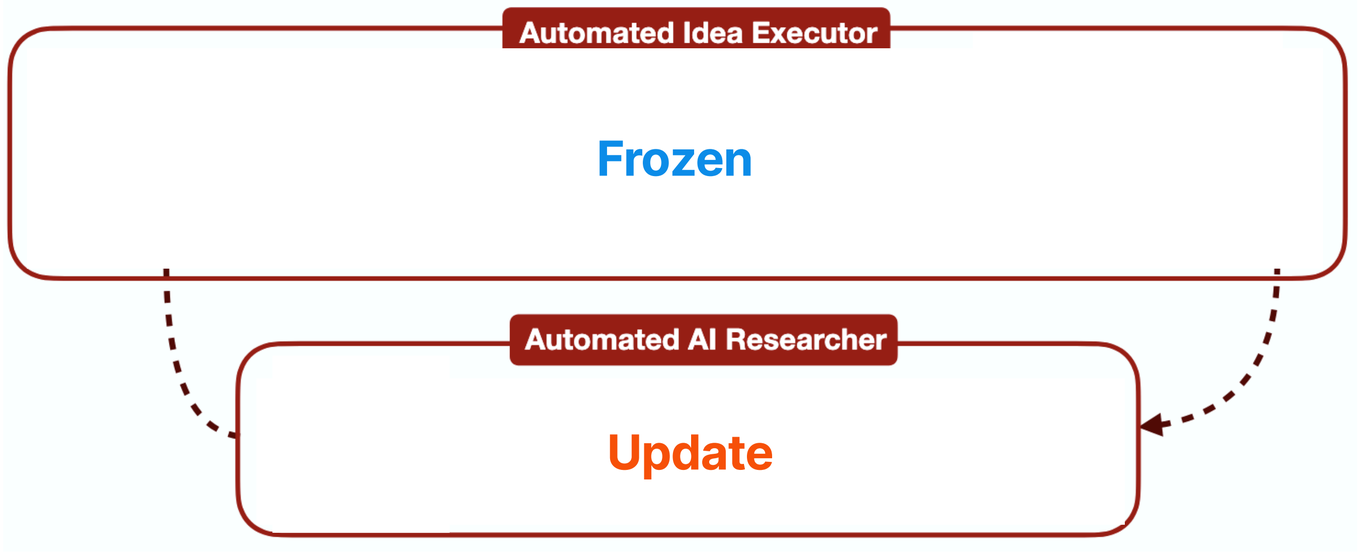
Domain	Conferences
NLP	ACL, EMNLP, NAACL, COLM
ML	NeurIPS, ICLR
CV	CVPR, ECCV, ACMMM
Robotics	CoRL

Table 2: Data statistics.

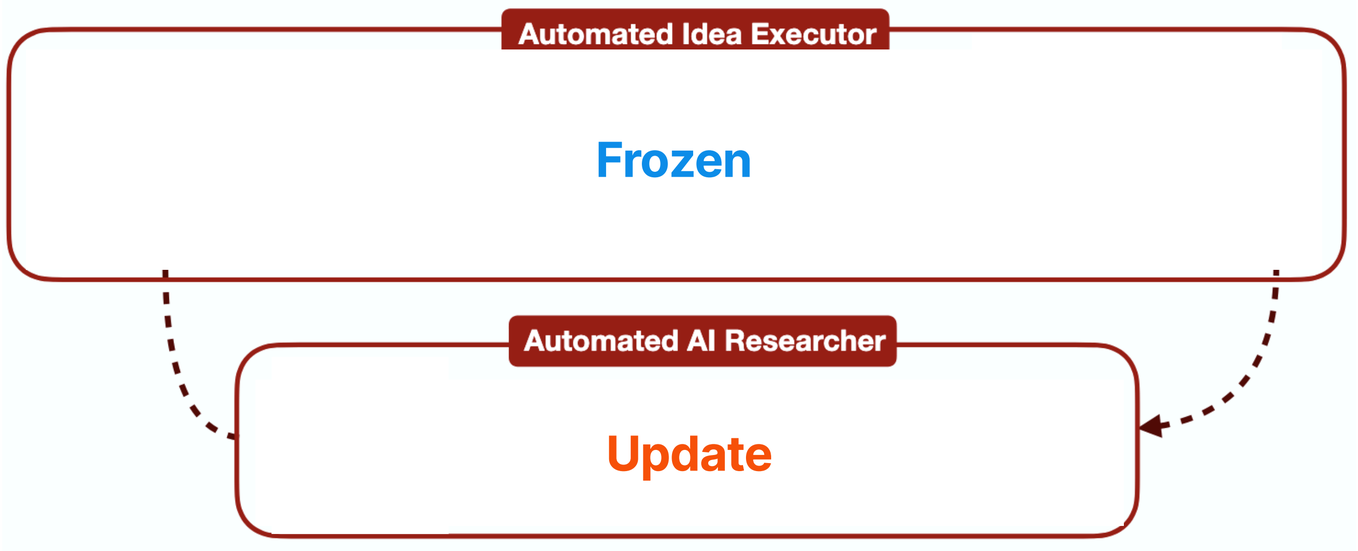
Split	Train	Test
Size	6,000	1,585
# Benchmark	3.2	3.4
After Cut-off	No	Yes
Human-Verified	No	Yes

Predicting Empirical AI Research Outcomes with Language Models, NeurIPS 2025

Let’s execute all the ideas to obtain the execution reward



How to do execution grounding is non-trivial



→ Can we get reward signal on idea effectiveness automatically?

→ Can the ideator models learn from the execution feedback?

We pick two realistic research environments

Pre-Training Task:

- Train Qwen2.5-Math-1.5B on MATH

Metric:

- Validation Accuracy

Baseline:

- Standard GRPO w/ clipping
- Acc = 48%

Post-Training Task:

- Train GPT-2 (124M) on FineWeb

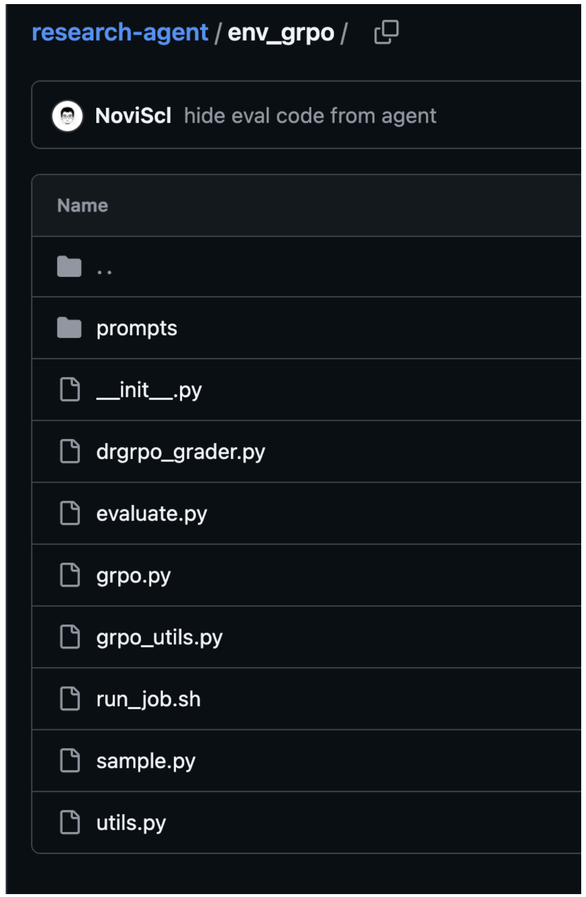
Metric:

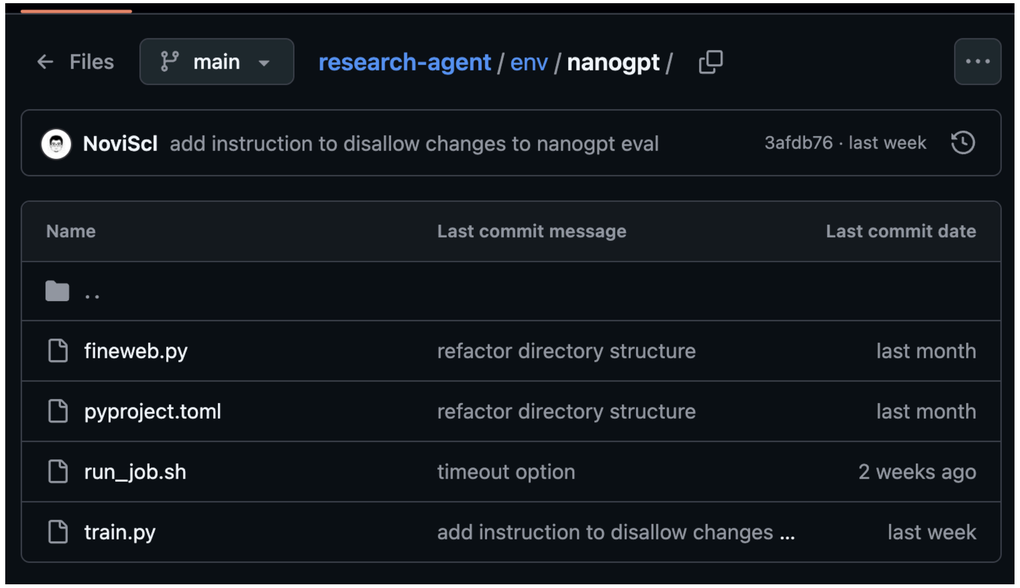
- Time to reach <=3.28 val loss on 8xH100
- Proxy metric: Val loss after training for 25min on 8xB200

Baseline:

- Adapted from Andrej Karpathy's nanoGPT
- Takes 35.9min to reach 3.28 val loss on 8xH100
- Val loss = 3.255 after 25min on 8xB200

We pick two realistic research environments





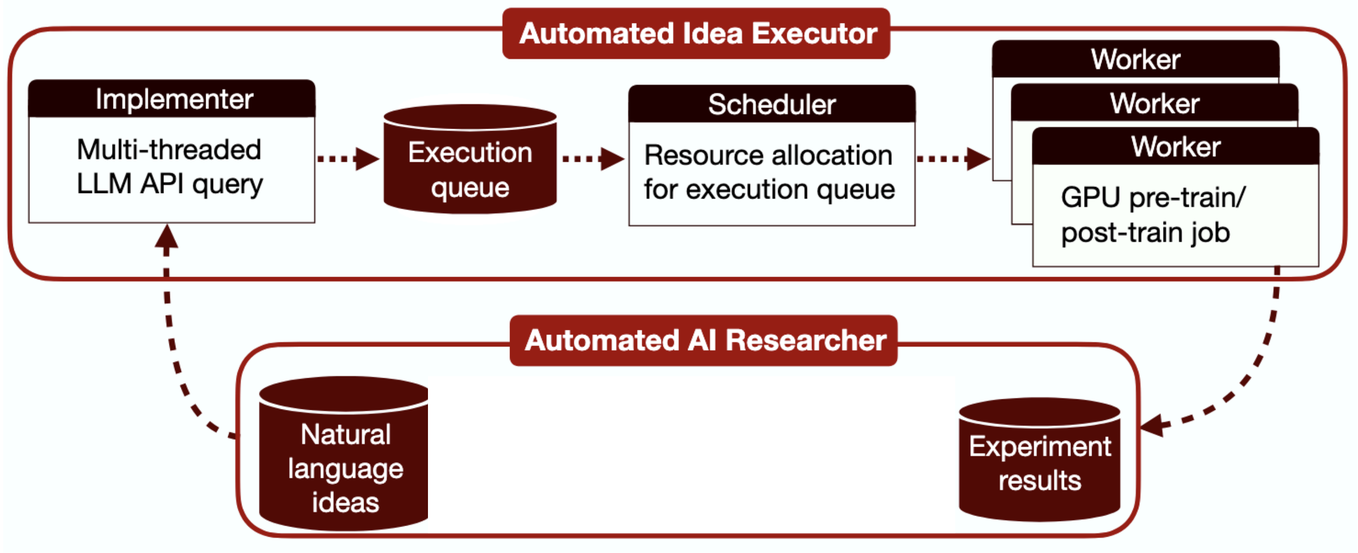
https://github.com/NoviScl/Automated-AI-Researcher

Scope: Open-Ended Research Ideas

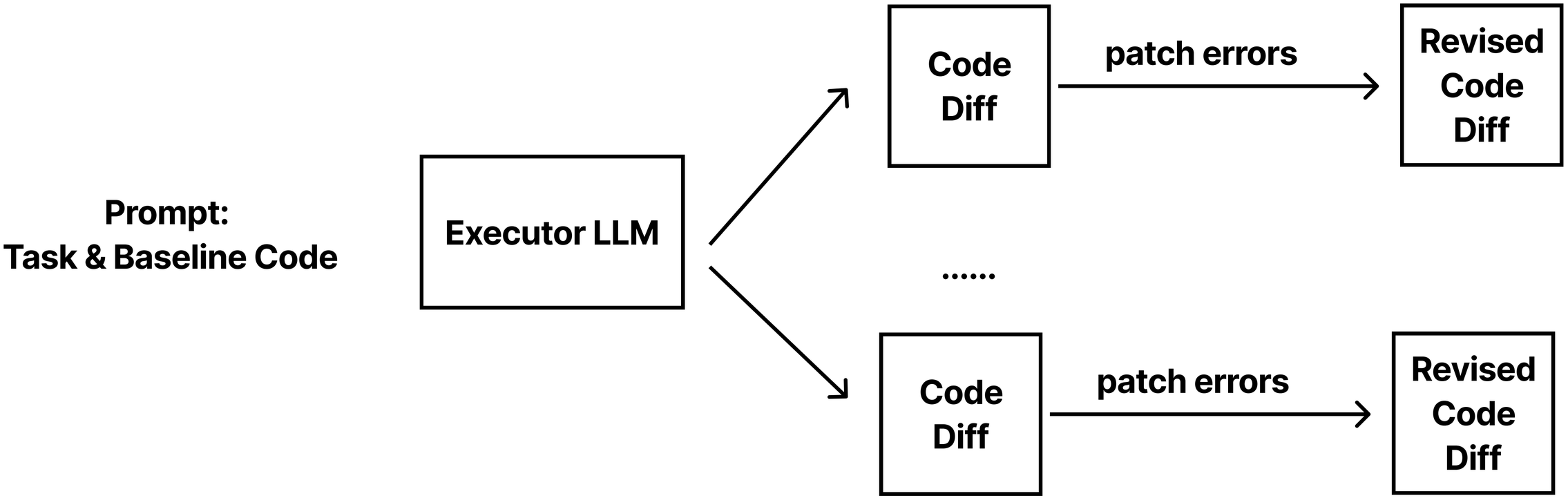
Claude-4.5-Sonnet on GRPO
Dynamic Mathematical Problem Difficulty Balancing with Performance Feedback: Implement intelligent difficulty balancing that dynamically adjusts the mix of problem difficulties based on recent performance trends. When performance is strong, increase difficulty proportion; when struggling, provide more foundational problems. Combine with the proven hyper-parameters for optimal learning progression. **Validation Accuracy: 64.0**
Implement token-level reward attribution by using attention weights to identify which input tokens contributed most to correct answers, then amplifying the gradient updates for those tokens during policy gradient training. **Validation Accuracy: 45.2**

- **Soft Layer Repetition** Allow the model to softly repeat computation through layers by adding a learned gate that mixes the current layer's input back into its output, simulating variable depth.
- **Causal Context Compression** Before each attention layer, apply a learned compression that mixes local context (previous 2-3 tokens) into the current representation, providing implicit local context without convolutions.
- **Attention Head Specialization via Orthogonal Loss** Add a soft penalty that encourages different attention heads to attend to different patterns by penalizing similarity between head outputs.
- **Skip Connections with Learned Residual Weights** Combine skip connections with learned residual weights. The skip connections provide alternative gradient paths while learned weights allow adaptive scaling.
- **Token Difficulty-Aware Loss Weighting** Weight the loss contribution of each token based on the model's uncertainty (entropy) at that position, focusing learning on difficult tokens while not over-optimizing easy ones.

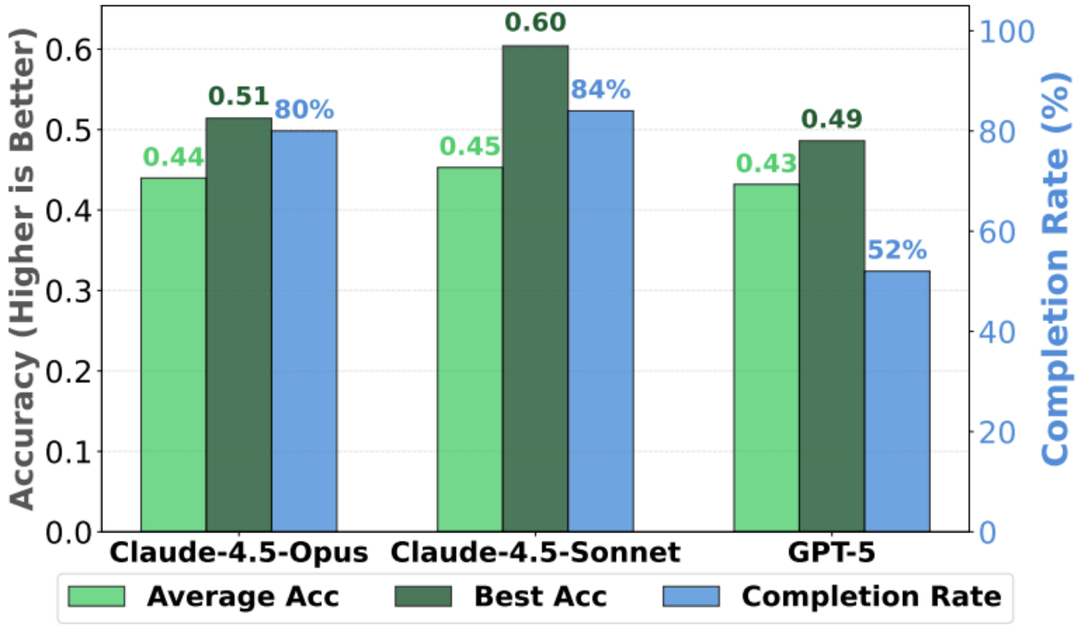
Building an automated idea executor -- System Design

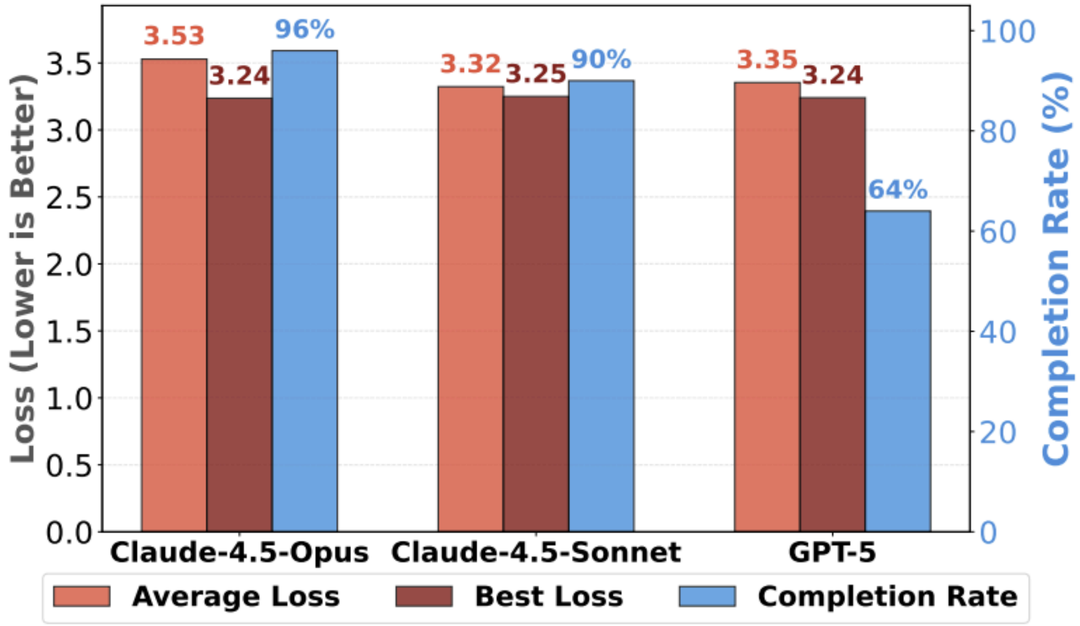


Building an automated idea executor -- System Design



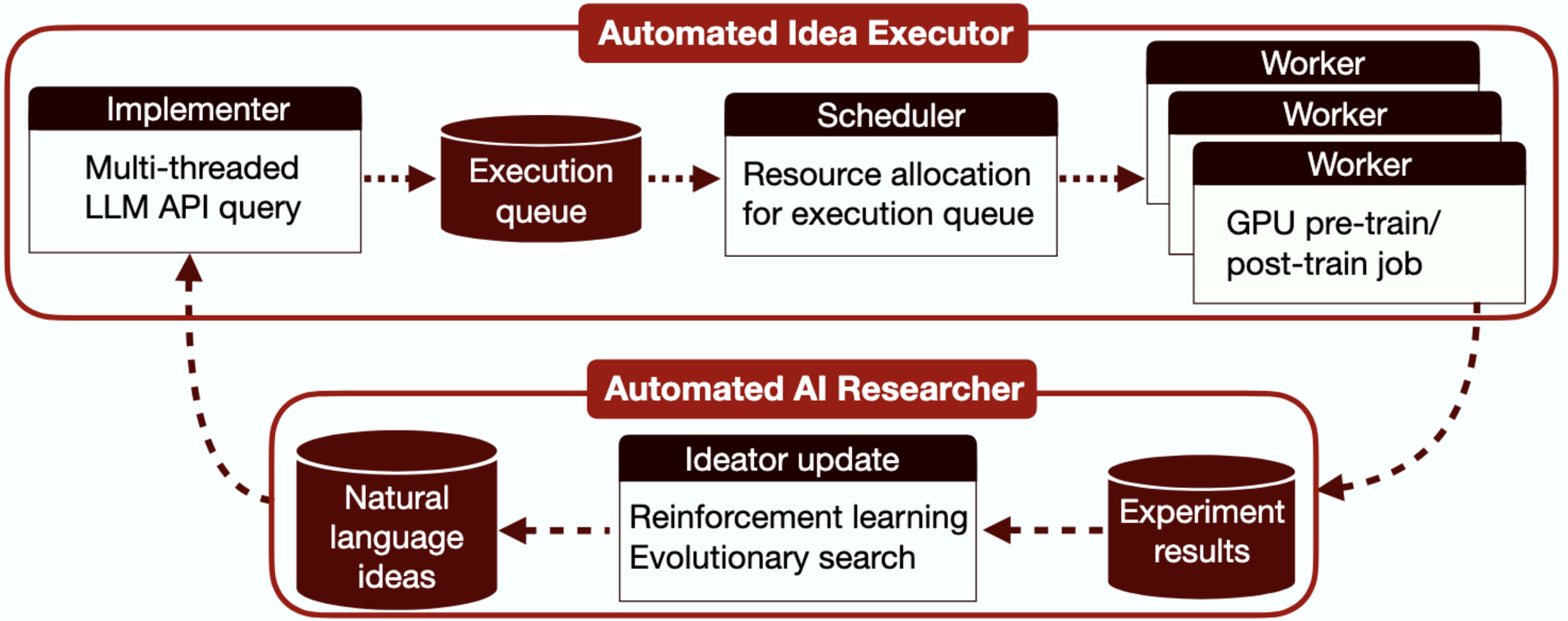
Does the automated executor work?





- **Execution rates are pretty high!**
- **Claude is better than GPT-5.**
- **Best-of-N (N=50) already beats the baselines!**

Benchmarking two well-established learning algorithms



Algorithm: Evolutionary Search

Algorithm 1 Execution-Guided Search

Require: batch size N, epochs T, baseline performance β
Require: initial exploitation rate $a_1 \in [0, 100]$, annealing schedule $a(t)$ for $t \in \{1, \ldots, T\}$

Sample initial batch of ideas $\mathcal{I}_0 \leftarrow \textsc{SampleIdeas}(N)$
Execute $\mathcal{I}_0$ to obtain trajectories $\mathcal{D}_0 \leftarrow \{(\text{idea}, \text{reward})\}$
for $t = 1$ **to** T **do**
  $a \leftarrow a(t)$ ▷ $(100 - a)\%$ exploration rate
  $\mathcal{D}_{<t} \leftarrow \bigcup_{k=0}^{t-1} \mathcal{D}_k$
  $\mathcal{D}^+ \leftarrow \{(i, r) \in \mathcal{D}_{<t} \ : \ r > \beta\}$ ▷ positive trajectories
  $N_{\text{exp}} \leftarrow \left\lfloor \frac{a}{100} N \right\rfloor, \quad N_{\text{expl}} \leftarrow N - N_{\text{exp}}$
  $\mathcal{I}_t^{\text{exp}} \leftarrow \textsc{ExploitVariants}(\mathcal{D}^+, N_{\text{exp}})$
  $\tilde{\mathcal{D}}_{<t} \leftarrow \textsc{SubsampleToContext}(\mathcal{D}_{<t})$
  $\mathcal{I}_t^{\text{expl}} \leftarrow \textsc{ExploreNovel}(\tilde{\mathcal{D}}_{<t}, N_{\text{expl}})$
  $\mathcal{I}_t \leftarrow \mathcal{I}_t^{\text{exp}} \cup \mathcal{I}_t^{\text{expl}}$
  Execute $\mathcal{I}_t$ to obtain trajectories $\mathcal{D}_t \leftarrow \{(\text{idea}, \text{reward})\}$
end for
return $\bigcup_{t=0}^{T} \mathcal{D}_t$

Google DeepMind

AlphaEvolve: A coding agent for scientific and algorithmic discovery

Alexander Novikov*, Ngân Vũ*, Marvin Eisenberger*, Emilien Dupont*, Po-Sen Huang*, Adam Zsolt Wagner*, Sergey Shirobokov*, Borislav Kozlovskii*, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli and Matej Balog*
Google DeepMind[1]

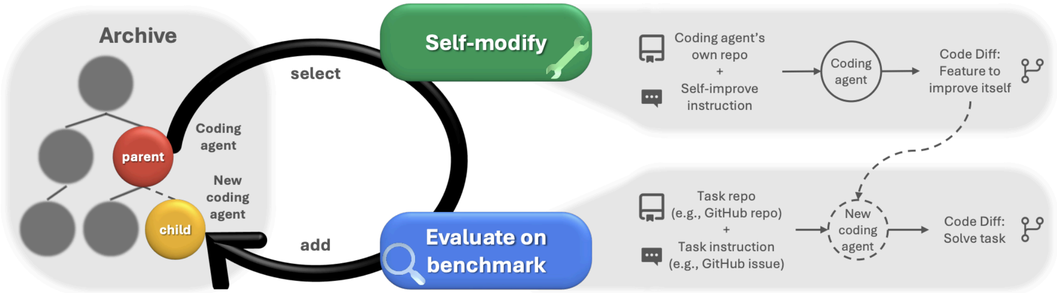
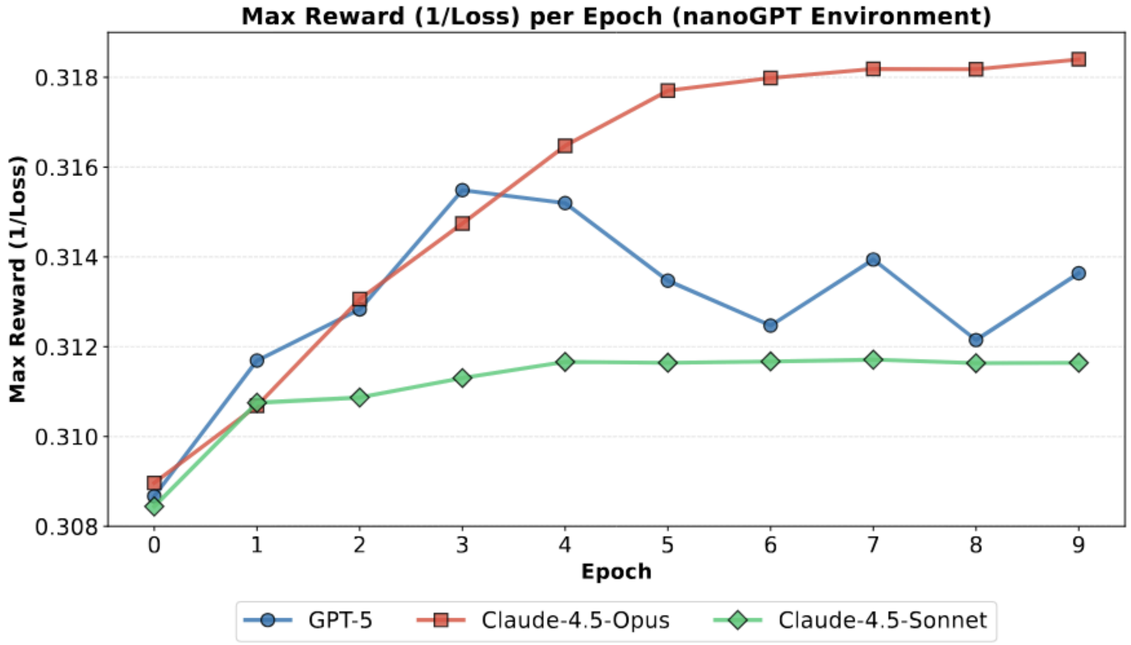
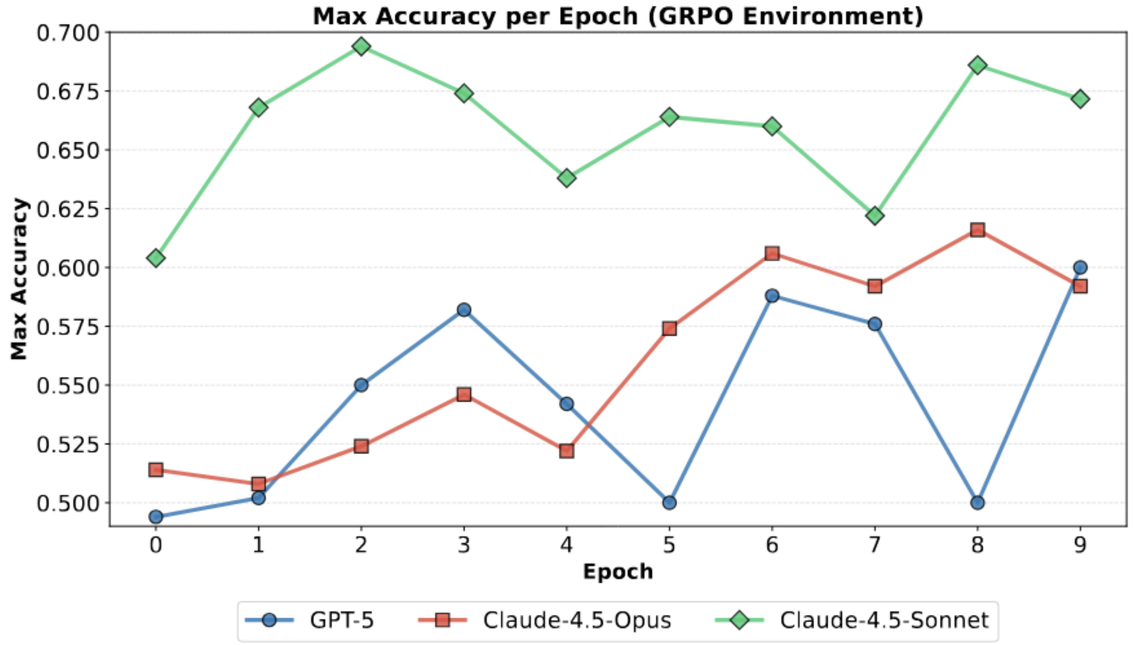


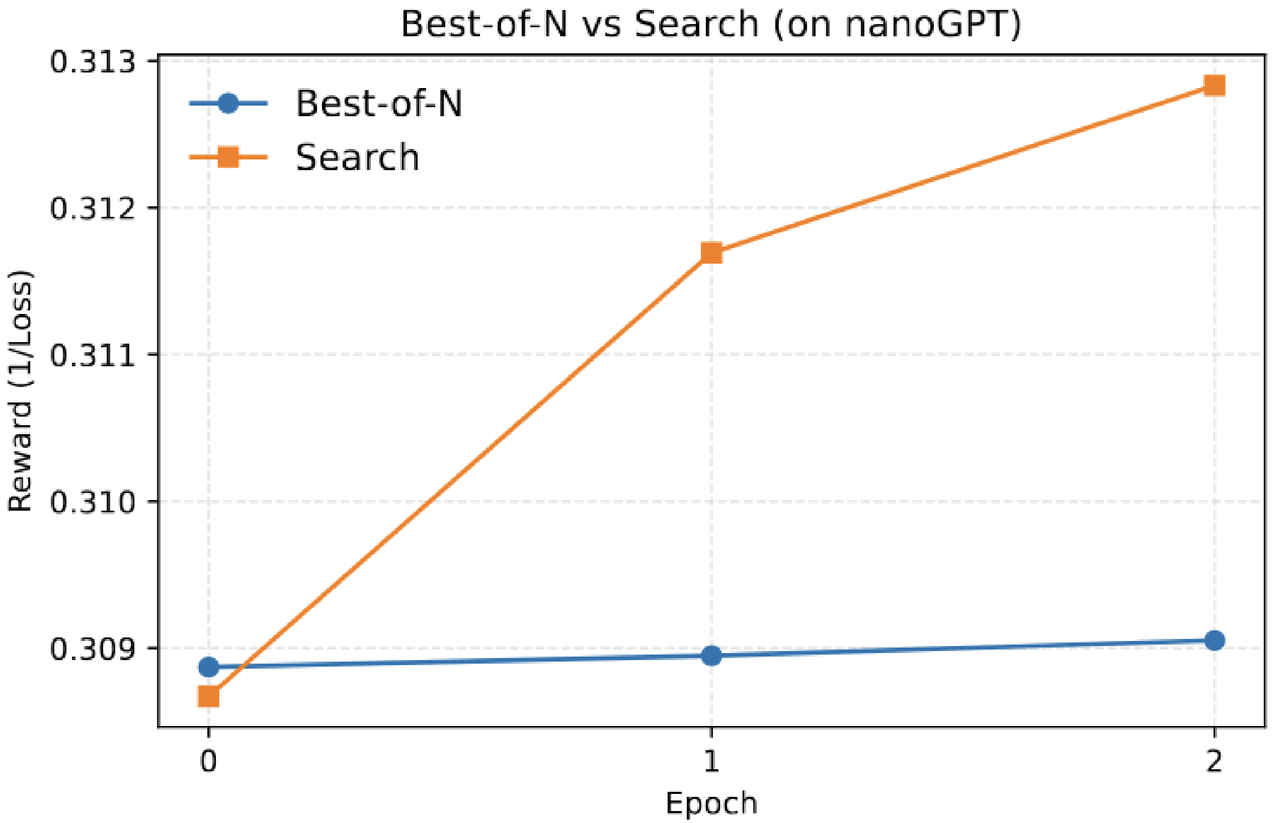
Figure 1: **Darwin Gödel Machine.** The DGM iteratively builds a growing archive of agents by interleaving self-modification with downstream task evaluation. Agents in the archive are selected for self-modification through open-ended exploration.

Algorithm: Evolutionary Search





Algorithm: Evolutionary Search



Algorithm: Evolutionary Search

Model name	Hyper-parameter			Algorithmic		
	Percentage	Average	Best	Percentage	Average	Best
GRPO environment (accuracy$_\uparrow$)						
GPT-5	5.0%	45.0%	50.2%	95.0%	44.5%	**60.0%**
Claude-4.5-Sonnet	41.1%	48.4%	**69.4%**	58.9%	45.0%	67.4%
Claude-4.5-Opus	3.7%	44.4%	50.4%	96.3%	46.5%	**61.6%**
nanoGPT environment (loss$_\downarrow$)						
GPT-5	15.4%	3.254	3.195	84.6%	3.894	**3.170**
Claude-4.5-Sonnet	31.3%	3.251	**3.208**	68.7%	3.679	**3.208**
Claude-4.5-Opus	8.7%	3.329	3.147	91.3%	3.419	**3.141**

Algorithm: Evolutionary Search

Claude-4.5-Opus on GRPO

Residual Ratio Learning with Momentum Bounds: Instead of directly using the (importance sampling) ratio, decompose it into a "base" component (EMA of batch mean ratios) and a "residual" component (ratio – base). Apply sigmoid bounding only to the residual, allowing the base to capture systematic policy drift while controlling deviations from it. Combined with momentum clip adaptation. Formula: `residual = ratio - ema_batch_ratio, bounded_residual = sigmoid_bound(residual, deviation),effective_ratio = 1.0 + bounded_residual.`

Validation Accuracy: 61.6

Advantage Rank Difference Weighting: Instead of using absolute advantage magnitude, weight samples by how far their rank differs from their expected rank under uniform distribution. Samples that significantly outperform or underperform their "expected" position get higher weights. This is distribution-free and robust to outliers. Formula: `expected_rank = (N-1)/2, rank_diff = |actual_rank - expected_rank| / expected_rank, weight = 0.5 + 0.5 * rank_diff.`

Validation Accuracy: 59.2

Algorithm: Evolutionary Search

Claude-4.5-Sonnet on GRPO

Dynamic Mathematical Problem Difficulty Balancing with Performance Feedback: Implement intelligent difficulty balancing that dynamically adjusts the mix of problem difficulties based on recent performance trends. When performance is strong, increase difficulty proportion; when struggling, provide more foundational problems. Combine with the proven hyper-parameters for optimal learning progression.

Validation Accuracy: 64.0

Create mathematical working memory simulation by maintaining a context buffer of mathematical facts, definitions, and intermediate results during problem solving. This buffer gets updated as the model works through problems and provides additional context for subsequent mathematical steps, simulating how humans maintain mathematical working memory during complex calculations.

Validation Accuracy: 58.0

GPT-5 on GRPO

Token-Level Ratio De-noising via Response Chunks (Chunked-Ratio): Reduce noisy token spikes by averaging log-ratio over small contiguous chunks within the response. Partition response tokens into C chunks per sequence (e.g., $C = 8$ over effective length), replace per-token $\Delta \log p$ with chunk mean broadcast to tokens in the chunk before ratio and clipping. Keeps structural signal while smoothing extremes.

Validation Accuracy: 58.2

Algorithm: Evolutionary Search

[Experiment] Implement response diversity rewards within groups where responses to the same prompt receive bonus rewards (`0.05--0.15`) for being dissimilar to other responses in their group, encouraging exploration of different solution paths while maintaining the proven `group_size=8` and `3e-5` learning rate combination.

[Code Changes] Modify `compute_group_normalized_rewards` in `grpo_utils.py` to compute pairwise similarity between responses in each group using token-level Jaccard similarity. Add diversity bonus: `diversity_reward = 0.15 * (1 - max_similarity_in_group)` to each response's reward before advantage computation. Reshape responses into groups, compute similarities, and add bonuses before advantage normalization. Set `--learning_rate 3e-5`, `--loss_type reinforce_with_baseline`, `--group_size 8`.

Validation Accuracy: 19.2

Jointly Reinforcing Diversity and Quality in Language Model Generations

Tianjian Li[♡◇†] **Yiming Zhang**[♡♣†] **Ping Yu**[♡] **Swarnadeep Saha**[♡] **Daniel Khashabi**[◇] **Jason Weston**[♡] **Jack Lanchantin**[♡] **Tianlu Wang**[♡]

[♡]Meta FAIR [♣]Carnegie Mellon University [◇]Johns Hopkins University
[†]Work done during an internship at Meta

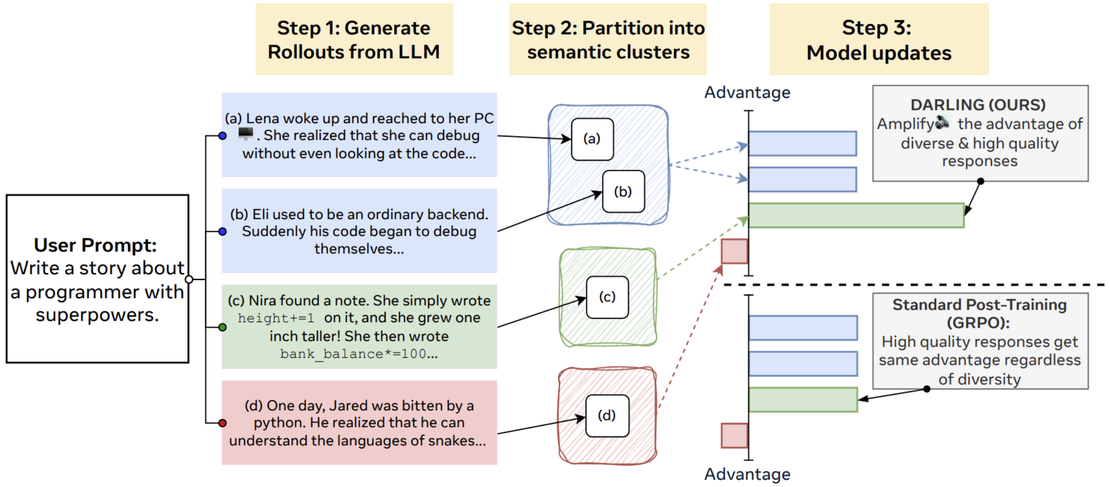


Figure 1 Diversity-Aware Reinforcement Learning (DARLING): We first partition LLM generations into semantically equivalent clusters (represented by colors). While standard GRPO (Shao et al., 2024) increases probabilities based on response quality only, DARLING amplifies the increase in probability of diverse and high-quality responses.

Algorithm: Evolutionary Search

Top-performing ideas usually involve extreme hyper-param tuning

Claude-4.5-Opus Idea on nanoGPT (Validation Loss: 3.1407)

[Experiment] Wider SwiGLU (5x) with MLP Output Scaling (Init 0.97), Skip Connections Every 4 and 8 Layers with Learnable Weights (Init 0.52 and 0.31), Separate Attention/MLP Scales (Init 0.98), Higher LR (0.00168), Reduced Weight Decay (0.065), Warmup 173 iters, Lower Min LR (0.03x), Cosine Annealing, EMA, Untied Embeddings, and Beta2=0.99
Make the dual skip connection weights learnable parameters initialized at proven good values. This allows the model to adapt skip weights during training while combining with separate attention/MLP residual scales.

Claude-4.5-Sonnet Idea on nanoGPT (Validation Loss: 3.2081)

[Experiment] Two-phase weight decay (0.1170→0.0210 at 59.65%) + 30.45% plateau + LR 0.001550 + warmup 197 + two-phase grad clip (1.054→0.916 at 59.65%) + quadratic min_lr interpolation (0.0113x at 59.65%, 0.0075x at end via quadratic) + progressive EMA (0.999→0.9992 linear over training) + exponential warmup + cosine LR + beta2=0.99

Algorithm: Evolutionary Search

Nevertheless, here are some interesting “atomic” ideas

Examples from Claude-4.5-Opus

- **Head-Wise Attention Output Scaling** Add learnable per-head scaling factors to attention, allowing different heads to contribute with different magnitudes to the output.
  Validation Loss: 3.2386
- **Learned Residual Connection Weights** Add learnable scalar weights for each residual connection that are initialized to 1.0, allowing the model to learn optimal residual scaling during training.
  Validation Loss: 3.2517
- **Mixture of Embeddings with Position** Learn to mix token embeddings and position embeddings with a content-dependent weight, allowing the model to dynamically balance positional vs semantic information per token.
  Validation Loss: 3.2497
- **Shared Input-Output Embedding with Learned Asymmetry** Keep weight tying but add a small learned transformation on the output side, providing the benefits of weight tying while allowing output-specific adaptation.
  Validation Loss: 3.2499
- **Gated Final Normalization** Replace the final RMSNorm before `lm_head` with a gated version where a learned gate controls how much normalization is applied vs passing the raw representation.
  Validation Loss: 3.2503
- **Position-Aware MLP Gating** Gate the MLP output based on position information, allowing the model to learn position-dependent processing depth.
  Validation Loss: 3.2506
- **Learned Residual Connection Weights** Add learnable scalar weights for each residual connection that are initialized to 1.0, allowing the model to learn optimal residual scaling during training.
  Validation Loss: 3.2517
- **Grouped Token Embeddings** Group the vocabulary into clusters and add a learned embedding per cluster on top of token embeddings, providing hierarchical vocabulary structure.
  Validation Loss: 3.2521

Algorithm: Evolutionary Search

Nevertheless, here are some interesting “atomic” ideas

Lastly, we present several interesting ideas on the nanoGPT environment that didn’t get successfully executed. These examples are generated by Claude-4.5-Opus.

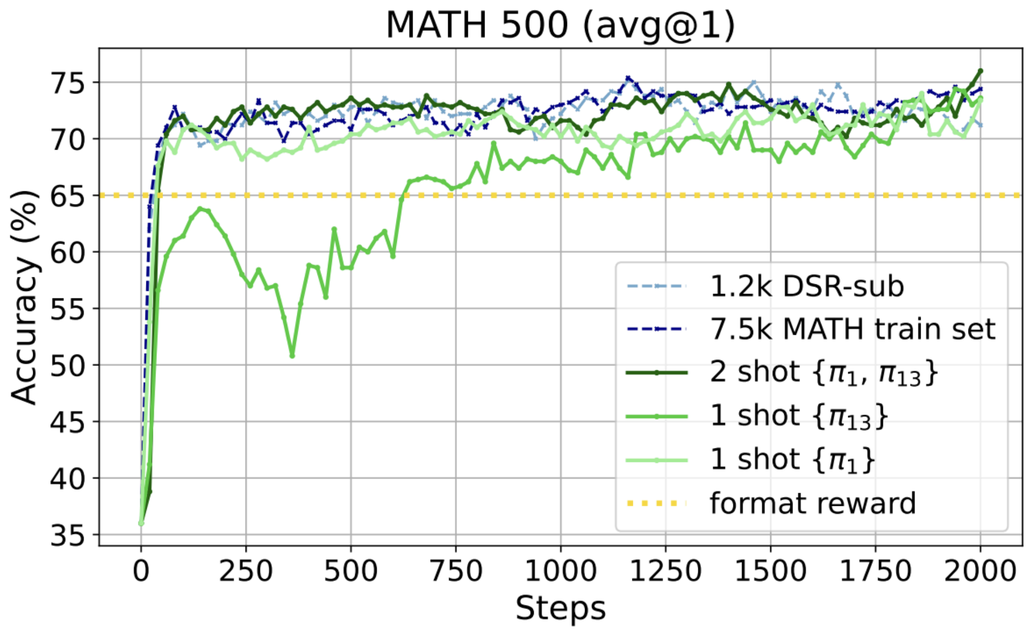
- **Soft Layer Repetition** Allow the model to softly repeat computation through layers by adding a learned gate that mixes the current layer’s input back into its output, simulating variable depth.
- **Causal Context Compression** Before each attention layer, apply a learned compression that mixes local context (previous 2-3 tokens) into the current representation, providing implicit local context without convolutions.
- **Attention Head Specialization via Orthogonal Loss** Add a soft penalty that encourages different attention heads to attend to different patterns by penalizing similarity between head outputs.
- **Skip Connections with Learned Residual Weights** Combine skip connections with learned residual weights. The skip connections provide alternative gradient paths while learned weights allow adaptive scaling.
- **Token Difficulty-Aware Loss Weighting** Weight the loss contribution of each token based on the model’s uncertainty (entropy) at that position, focusing learning on difficult tokens while not over-optimizing easy ones.

Algorithm: RL

Vanilla GRPO from DeepSeek

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$
$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}, \quad (3)$$

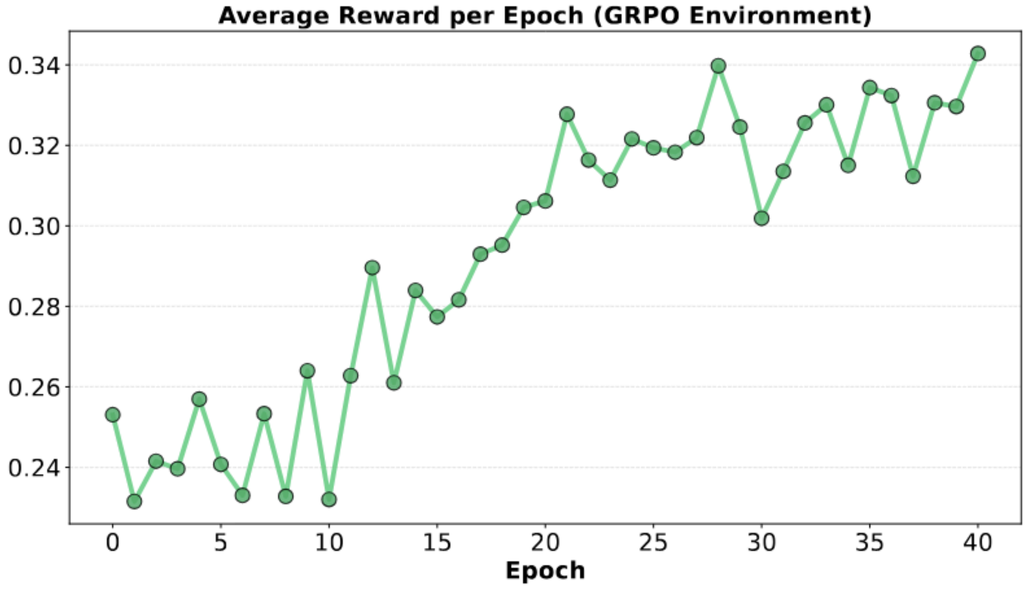
We only have one prompt for each environment, so essentially “one-shot RLVR”

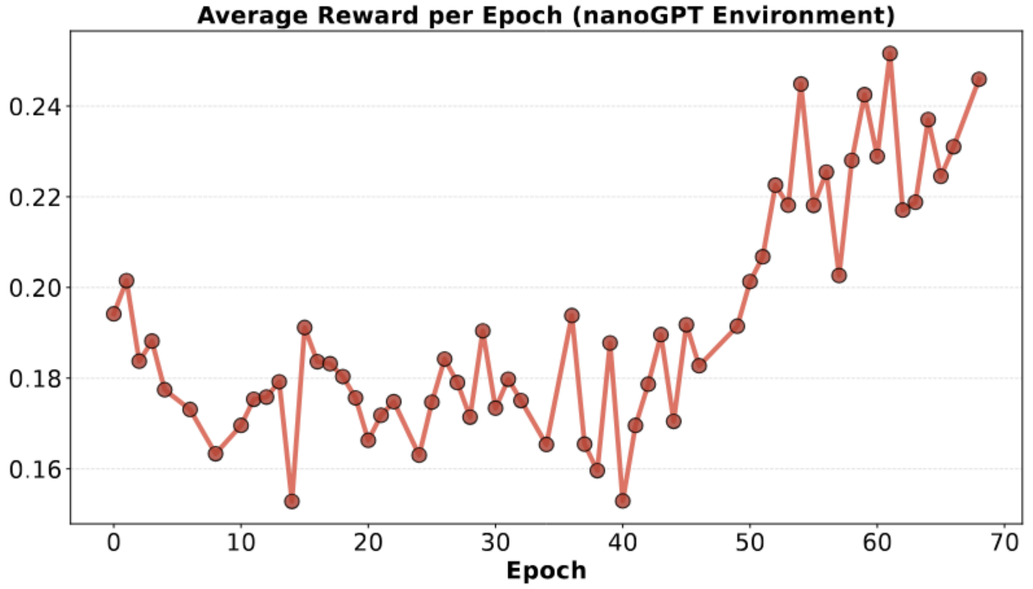


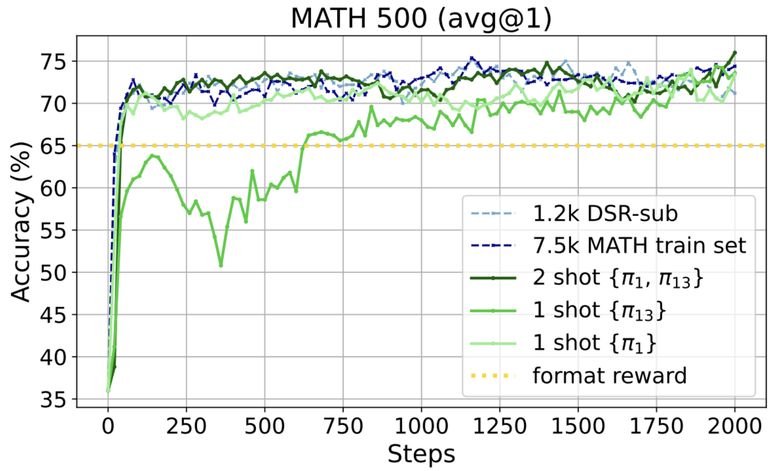
Reinforcement Learning for Reasoning in Large Language Models with *One* Training Example

Yiping Wang[1 † *] Qing Yang[2] Zhiyuan Zeng[1] Liliang Ren[3] Liyuan Liu[3]
Baolin Peng[3] Hao Cheng[3] Xuehai He[4] Kuan Wang[5] Jianfeng Gao[3]
Weizhu Chen[3] Shuohang Wang[3 †] Simon Shaolei Du[1 †] Yelong Shen[3 †]
[1]University of Washington [2]University of Southern California [3]Microsoft
[4]University of California, Santa Cruz [5]Georgia Institute of Technology

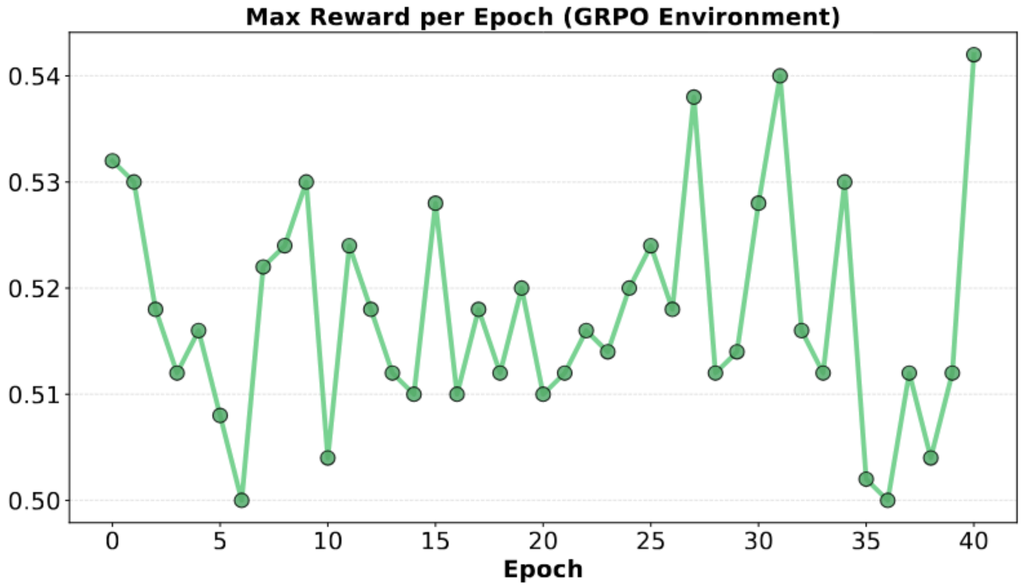
Algorithm: RL

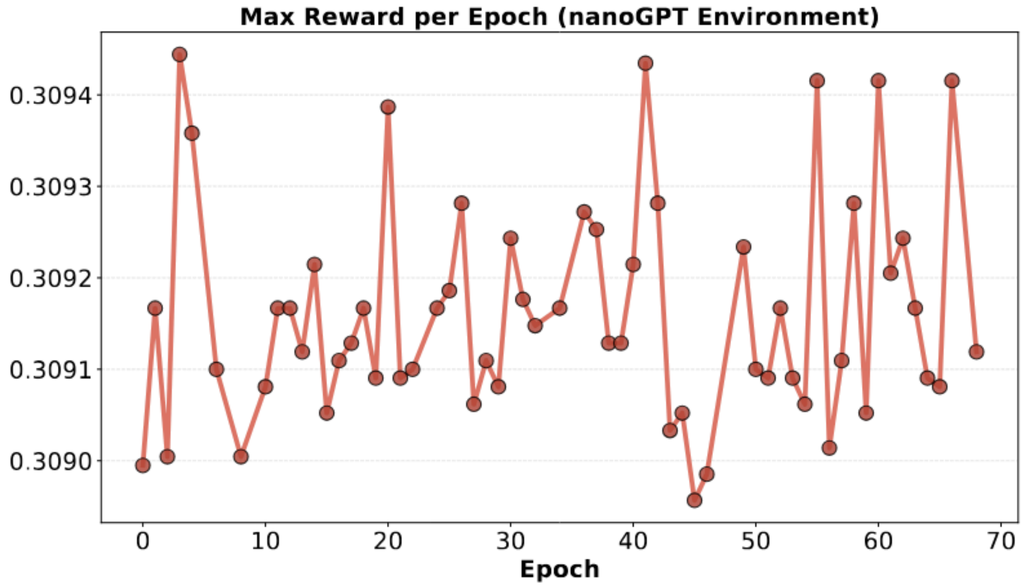




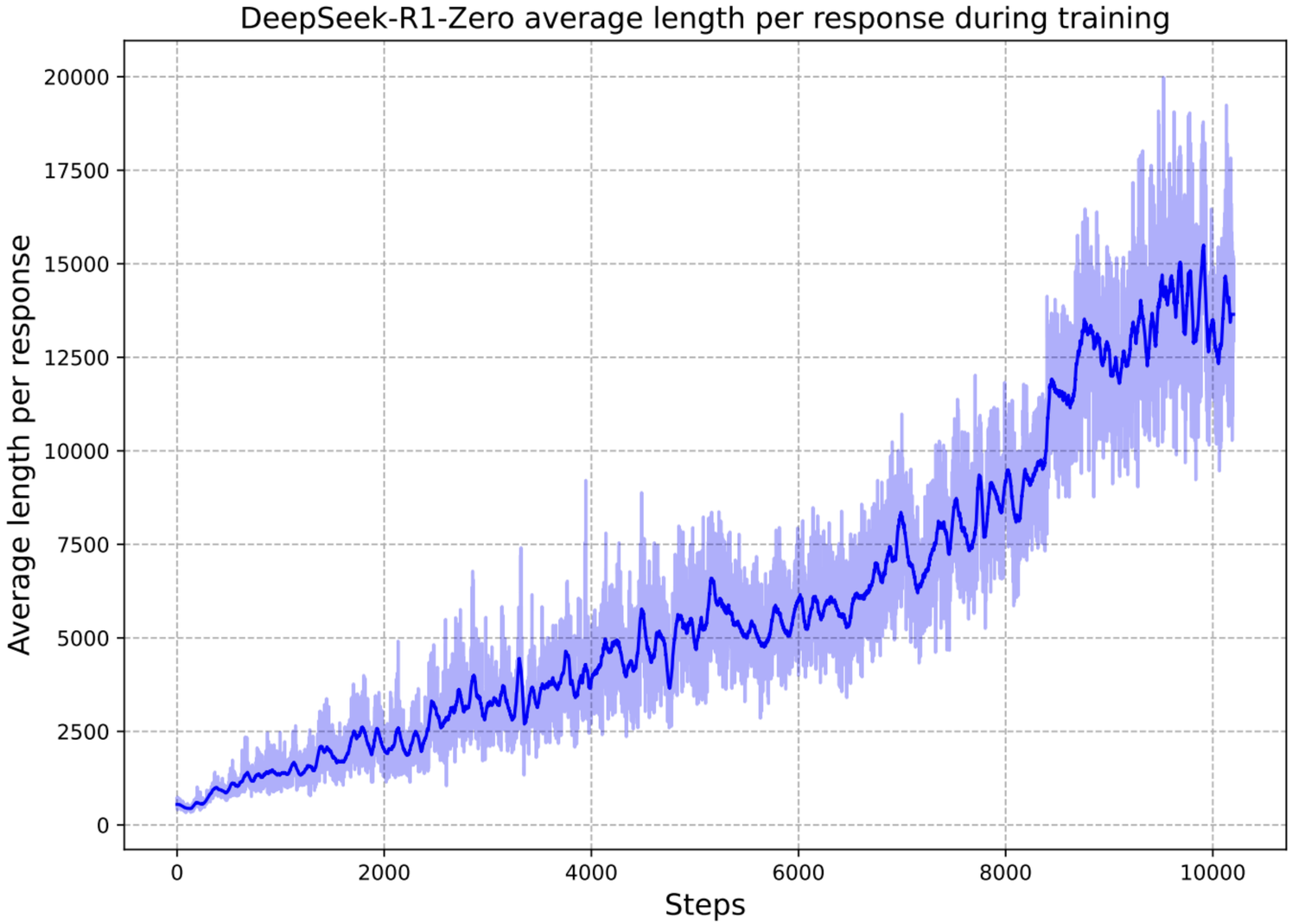


Algorithm: RL

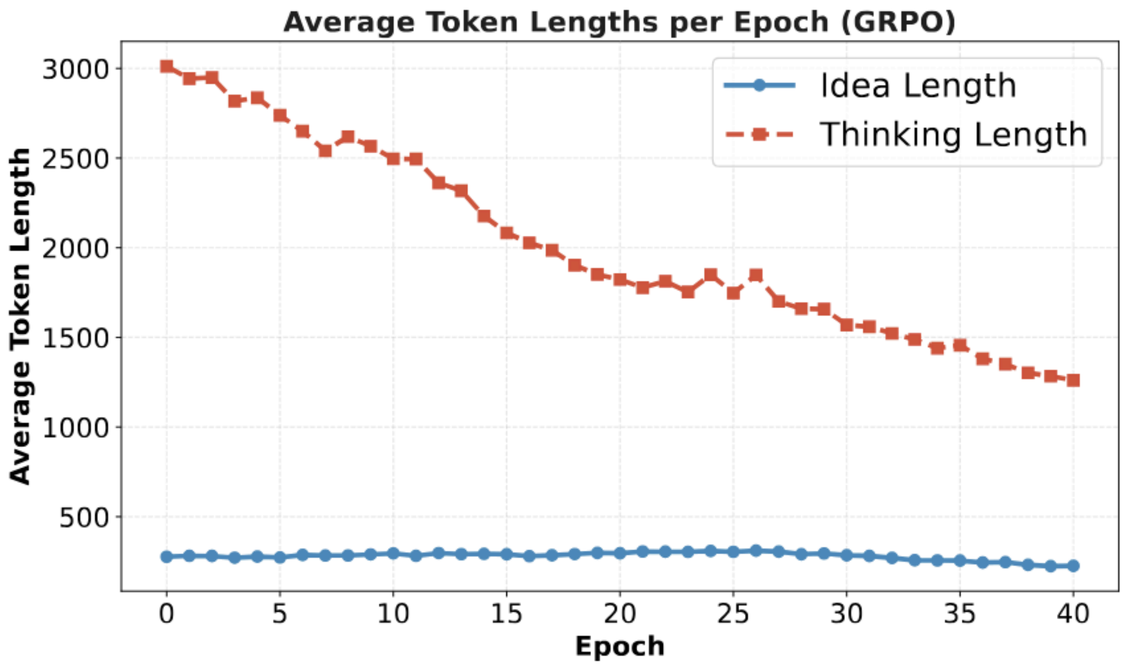


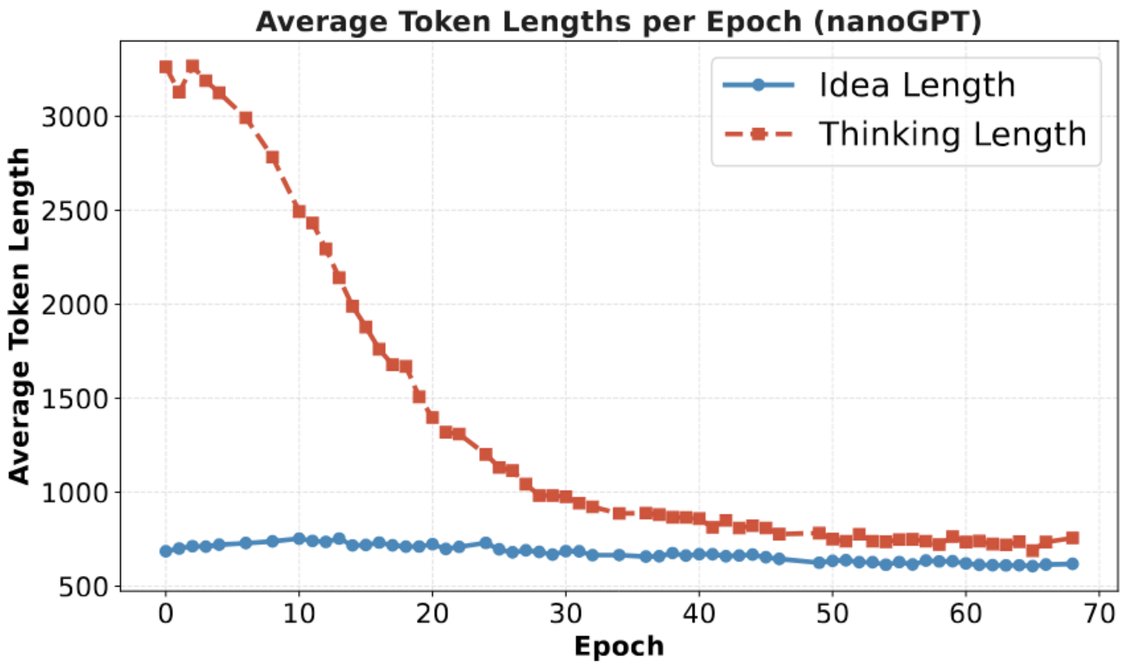


Algorithm: RL

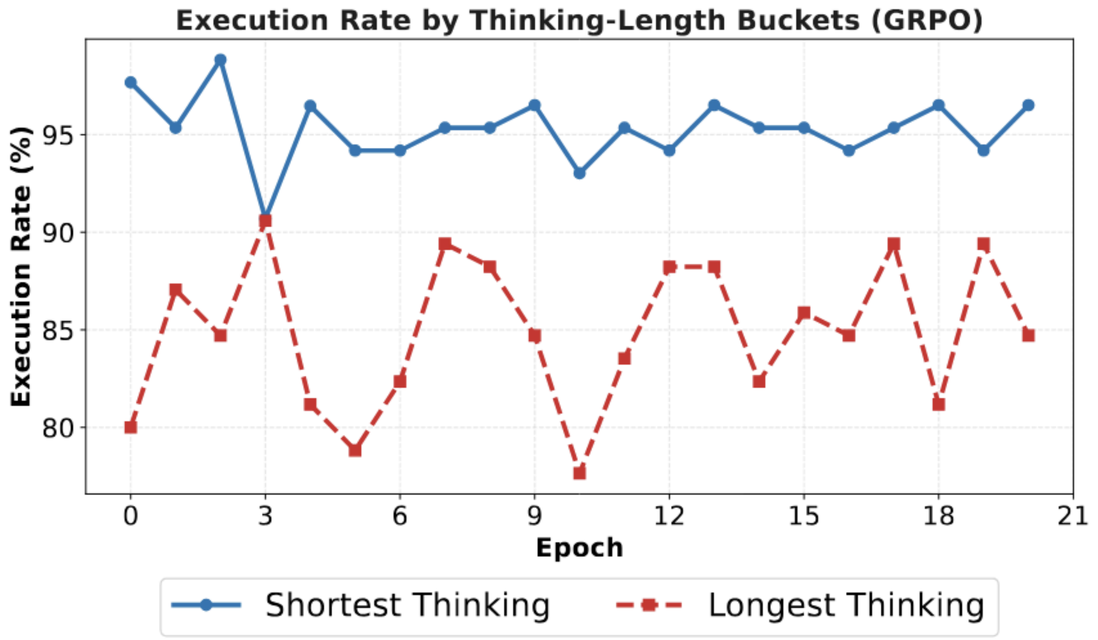


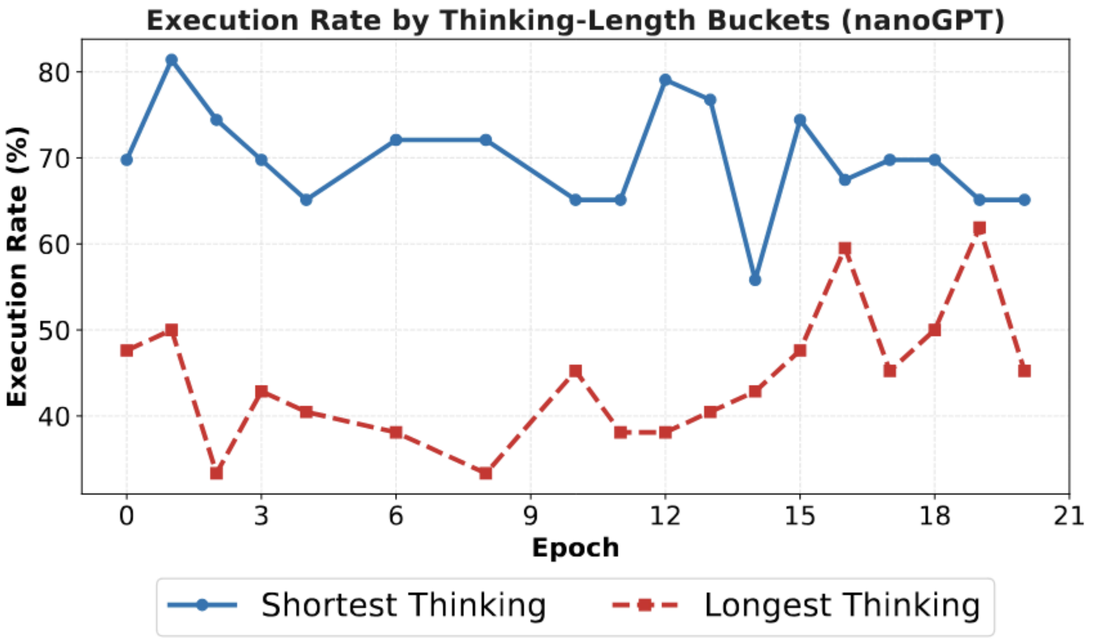
Algorithm: RL



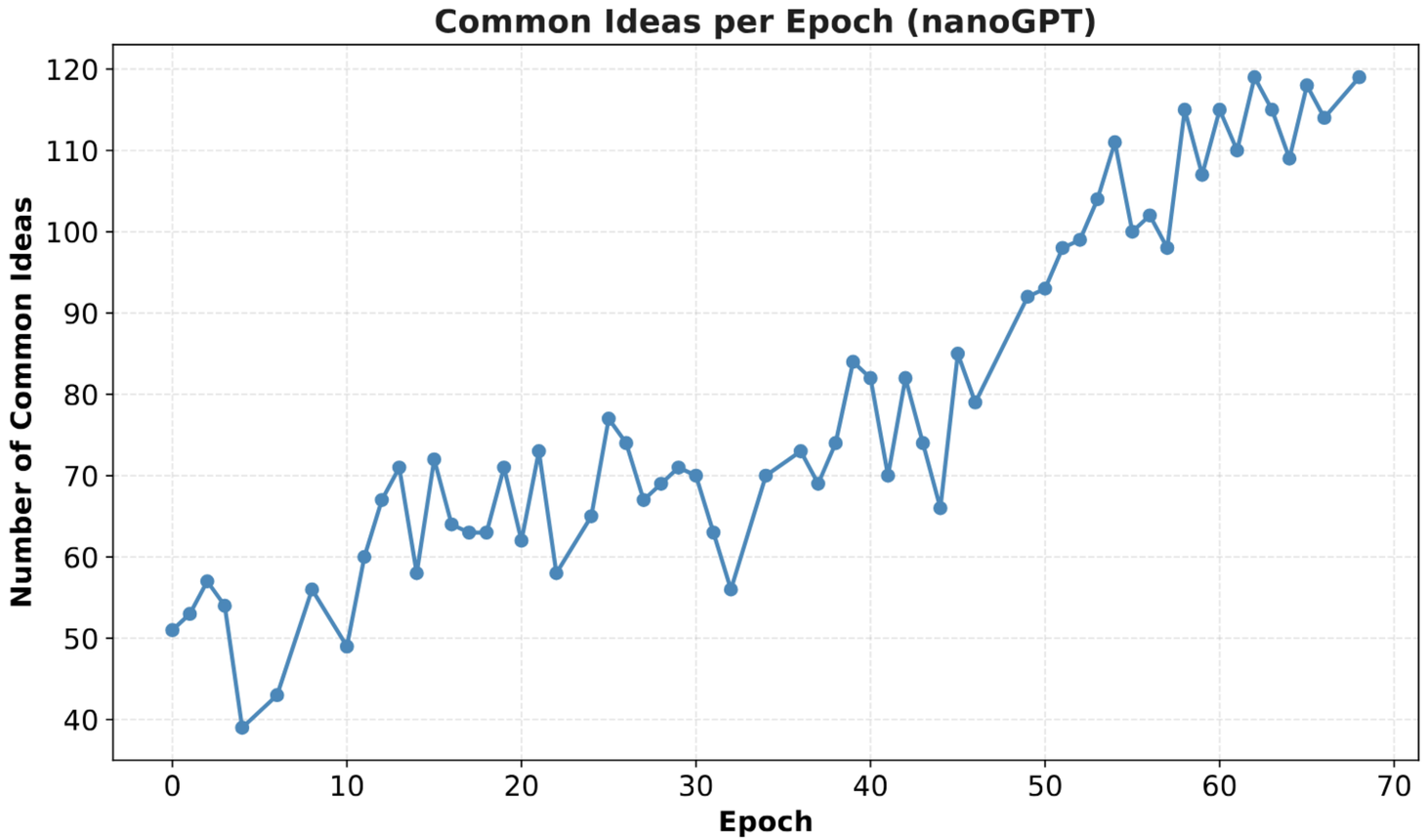


Algorithm: RL





Algorithm: RL



Common Ideas on nanoGPT:

- **EMA**
- **RMSNorm ↔ LayerNorm**

Summary

- **We demonstrated that execution grounding is *feasible*: automated executor can now implement a substantial number of open-ended ideas.**
- **The automated execution feedback can already guide evolutionary search to find more effective ideas (including both hyper-parameter and algorithmic ideas).**
- **RL with execution reward can reproduce the typical pass@1 training curve of RLVR, but pass@k doesn’t go up, because models converged on a few easy ideas.**

Future Directions

- **Scale: Most ideas at the 1.5B / 124M scale probably won’t scale. Ultimately we care about ideas that can scale.**
- **Learning Algorithm: We should probably leverage more information than just a scalar reward (we spent $$$ to score each rollout, we should try to get more out of it).**
- **Reward: Beyond just chasing for effectiveness, some of the most impactful research is about discovering interesting insights. We have no reward for that yet.**

Can LLMs Generate Novel Research Ideas?

A Large-Scale Human Study with 100+ NLP Researchers

Chenglei Si, Diyi Yang, Tatsunori Hashimoto
Stanford University
{clsi, diyiy, thashim}@stanford.edu

https://arxiv.org/abs/2409.04109

The Ideation–Execution Gap: Execution Outcomes of LLM-Generated versus Human Research Ideas

Chenglei Si, Tatsunori Hashimoto, Diyi Yang
Stanford University
{clsi, thashim,diyiy}@stanford.edu

https://arxiv.org/abs/2506.20803

Predicting Empirical AI Research Outcomes with Language Models

Jiaxin Wen[1], **Chenglei Si**[2], **Chen Yueh-Han**[3], **He He**[3], **Shi Feng**[4]

[1]UC Berkeley [2]Stanford [3]New York University, [4]George Washington University
jiaxin.wen@berkeley.edu, shi.feng@gwu.edu

https://arxiv.org/abs/2506.00794

Towards Execution-Grounded Automated AI Research

Chenglei Si [*1] **Zitong Yang** [*1] **Yejin Choi** [1] **Emmanuel Candès** [1] **Diyi Yang** [1] **Tatsunori Hashimoto** [1]

https://arxiv.org/abs/2601.14525