

Talk that walk: Zero-shot LLM Curriculum Generation with Chain of Thought Models

Adam Vawda-Oomerjee

Federica Dani

Nora Burkhardt

Gianluca Carrozzo

Abstract

Curriculum learning can significantly improve training efficiency in reinforcement learning, but designing effective curricula typically requires domain expertise. This study investigates the zero-shot capabilities of two open-source LLMs —DeepSeek-V3 and DeepSeek-R1—for automatically generating executable, multi-stage reward functions for the Gymnasium BipedalWalker environment. Without iterative refinement or human feedback, both models produced viable curricula, with results at least on par with the human-designed curriculum. However, analysis of learning dynamics revealed performance variability, including regressions linked to misaligned rewards or flawed task sequencing. These results highlight the potential of LLMs for automated curriculum generation, while also exposing the challenges of consistency and reliability in zero-shot settings.

1 Introduction

Deep reinforcement learning (RL) has achieved remarkable success across various domains, including simulated environments, games, and robotics tasks. However, training RL agents from scratch, starting with random policies, often suffers from significant sample inefficiency, hindering the learning of complex tasks (Silver et al., 2017). Inspired by developmental strategies observed in human learning, recent RL research emphasizes curriculum learning, where agents are guided through a logically structured sequence of tasks increasing in complexity (Narvekar et al., 2020). This method improves training efficiency by initially concentrating on simpler tasks and progressively transferring acquired knowledge to more challenging scenarios (Laz; Narvekar et al., 2020). Despite these advantages, designing effective curricula typically demands considerable domain expertise, motivat-

ing interest in automated curriculum generation (autocurricula) (Li2; Tang et al., 2021; Tid).

Recent advancements in large language models (LLMs) have demonstrated their potential for autocurricula design and refinement, leveraging their enhanced reasoning and task-generalization capabilities (Wang et al., 2023a, 2024; Liang et al., 2024). Emerging methodologies combine the natural language processing strengths of LLMs with evolutionary algorithms, significantly improving the performance of RL models through more sophisticated reward design (Ryu et al., 2024; Hazra et al., 2024; Ma et al., 2023). Although several studies utilize LLM-generated curricula as baselines, explicit benchmarking of the zero-shot capabilities of LLMs for autonomous curriculum generation remains limited. Moreover, prior research primarily relies on GPT-4 models (Wang et al., 2023a; Ryu et al., 2024; Hazra et al., 2024; Ma et al., 2023), leaving unexplored the potential improvements in zero-shot curriculum generation from newer, RL-trained chain-of-thought models, such as DeepSeek-R1 and OpenAI’s o3.

Our research addresses this gap by benchmarking the zero-shot capabilities of DeepSeek’s open-source V3 and R1 models for autonomously generating executable reward-based curricula. Specifically, we evaluate their effectiveness in solving Gymnasium’s BipedalWalker environments, both standard and hardcore variants, through zero-shot prompting.

2 Related Literature

2.1 Curriculum Learning in RL

The use of curricula in reinforcement learning has unlocked major advances in training artificial agents to efficiently and effectively accomplish complex tasks (Wan). Early curriculum learning applications date back to the late 20th century, ap-

pearing in grammar construction, robotics, and classification problems (Elman; Ben). Recent research underscores its effectiveness in faster convergence and improved generalization (Wan), (Wang et al., 2023b), (Narvekar et al., 2020)). As a result, curriculum learning has exploded in the world of robotics and simulations. Despite these successes, optimally defining and structuring curricula generally relies on substantial domain knowledge and human intervention (Narvekar et al., 2020; Wan; Ryu et al., 2024).

2.2 LLMs for Advanced Tasks

Advancements in large language models, such as GPT-4 (Achiam et al., 2023), LLaMA (Touvron et al., 2023), and Gemini (Reid et al., 2024), have significantly expanded capabilities in automated reasoning and coding. Chain-of-thought prompting (Wei et al., 2022) and iterative self-refinement (Madaan et al., 2023) notably enhanced performance on complex reasoning benchmarks. Furthermore, Chen et al. illustrated the capabilities of LLMs explicitly trained for coding, highlighting their potential in automated code generation and optimization problems (Chen et al., 2021). These developments motivated the creation of reinforcement learning-based LLMs employing chain-of-thought methods, such as OpenAI’s o1 and DeepSeek-R1 (DeepSeek-AI and et al., 2025).

2.3 LLM-generated Rewards

Eureka pioneered the use of LLMs for reward design, achieving super human-level reward functions through iterative generation and greedily mutating the best candidate over multiple generations (Ma et al., 2023). Extending this approach, REvolve combined evolutionary mutation strategies with human feedback to enhance computational efficiency compared to Eureka’s “greedy” approach (Hazra et al., 2024). CurricuLLM further advanced this domain by automating curriculum design through a three-step process specifically tailored to complex robotic tasks without predefined evaluation metrics (Ryu et al., 2024). This method involved GPT-4 generating natural language subtasks, translating them into executable task codes, and evaluating trained policies. Additionally, Song et al. proposed a self-refined LLM framework for automated reward function design, showcasing promising developments in automating RL training (Song et al., 2023).

Voyager introduced an alternative, open-ended

approach leveraging LLMs for bottom-up curriculum generation, encouraging exploratory task discovery within embodied environments (Wang et al., 2023a). Unlike Voyager’s bottom-up skill acquisition, most LLM-driven curricula, including our research, adopt a top-down approach, aiming to systematically structure tasks toward a predefined goal. Additional research such as Auto MC-Reward (Li et al., 2024), Gensim (Wang et al., 2024), Prog-prompt (Singh et al.), and Text2Reward (Xie et al., 2024) further illustrate the adaptability and effectiveness of LLM-generated curricula and rewards across diverse simulation scenarios.

2.4 Comparison to Existing Methods

Distinct from previous studies, our research specifically evaluates zero-shot curricula and reward functions generated by DeepSeek-V3 and DeepSeek-R1 using few-shot prompting based on the default reward function provided by Gymnasium’s BipedalWalker environment. Unlike CurricuLLM, Eureka, or REvolve, our methodology excludes iterative human or evolutionary feedback loops, directly benchmarking LLM-generated outputs against human-designed curricula and baseline rewards.

3 Methods

3.1 Environment Choice

In selecting an environment, we considered both the compatibility of the target task with curriculum learning and the ease of accessing and replicating the environment. Although Eureka leverages IsaacGym environments, we chose to use Gymnasium BipedalWalker as it is less computationally expensive, the task is more simple to enable some success in a zero-shot setting, and the hardcore environment is difficult enough to justify the use of a curriculum.

3.2 Baselines

Our choice of baseline to compare the LLM-generated curricula includes both BipedalWalker’s standard reward with no curriculum and a generic human-generated curriculum which is representative of the skills of a master’s student with domain familiarity.

The human curriculum was designed based on domain knowledge of the BipedalWalker environment, drawing directly from the official Gymnasium documentation (Foundation, 2024) to understand its reward mechanics, termination conditions,

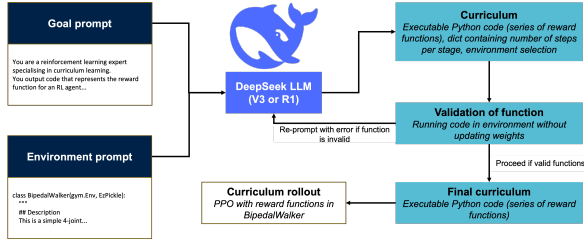


Figure 1: LLM Prompting Process

and terrain structure. To guide the progressive structure of the curriculum, we referenced foundational work on curriculum learning (Bengio et al., 2009), which supports the idea of training from simpler tasks (e.g., standing or walking on flat terrain) to more complex behaviors (e.g., traversing obstacles efficiently). Additional inspiration was taken from reward engineering practices in robotic locomotion tasks (Peng et al., 2020), motivating the inclusion of forward motion incentives, tilt penalties, and torque minimization as learning progresses.

3.3 LLM Prompting Process

LLM’s were prompted in a similar method to REvolve (Hazra et al., 2024), provided with a compact set of guidelines and hard constraints in natural language, a list of available and unavailable variables, and the structure of the reward function outputs (see Figure 1). To make the comparison between human and LLM curricula fair, we also gave the LLM access to BipedalWalker’s source code, notably how reward is ordinarily calculated by the environment (as we had access to this information when designing our curriculum). The LLM outputs the curriculum as a series of Python functions, a dict specifying the number of training steps each stage of the curriculum should train for, and a list specifying whether training should take place in hardcore or standard. Each function was tested using random actions in the environment to check validity. If a function was invalid, the error was captured and used to reprompt the LLM, whereby the LLM would return a function fixing the error.

3.4 RL algorithm

We opted to using Proximal Policy Optimisation (PPO) (Schulman et al., 2017) for training the agents once reward function, steps, and environment were chosen. PPO has been shown to provide good results for continuous control tasks like the ones presented in BipedalWalker (Aydogmus and Yilmaz, 2024) while being relatively computation-

ally cheap, which made it a natural choice for our study. We used an implementation of PPO adapted from CleanRL (Huang et al., 2022), an open source library that provides single file implementations of RL algorithms.

3.5 Evaluation

After training, policies are compared based on average distance traveled across 10 evaluation episodes.

4 Experiments

4.1 Trials

Experiments were conducted in three distinct setups wherein an agent was trained for 3 million steps and then evaluated over 10 episodes. The setups are as follows: (1) standard environment, which does not include obstacles, (2) hardcore environment which includes stairs, boxes and pitfalls and (3) mixed, where each stage of the curriculum consisted not only of the reward function and number training steps, but also which environment the agent is trained. For standard, the agent is evaluated in standard at the end of training. For hardcore and mixed, the agent is evaluated in hardcore. Both the human and LLM generated curricula worked within these constraints. For the no curriculum baseline, a single reward function (the default given by BipedalWalker) is used for training. In mixed, the agent trains for 1.5 million steps in standard, and then 1.5 million steps in hardcore. Evaluation environment is as before. For the LLM-generated curricula three identical trials of generating curricula, training and evaluating are conducted per environmental setup, resulting in distinct curricula which we compare in our results. All experiments were run with 32 environments in parallel using a single NVIDIA L40 GPU.

4.2 Model

An actor-critic model was trained using PPO. This is a model consisting of a “critic” network and an “actor” network, where the critic network estimates the value function of a given state and the actor network which determines the action to take. Hyperparameters were kept the same across all experiments, and were reset at the beginning of each curriculum stage. Model architecture was taken from CleanRL and full specifications can be found in the appendix. Environments in the CleanRL implementation use the Gymnasium “NormalizeObservation” wrapper, where the normalization con-

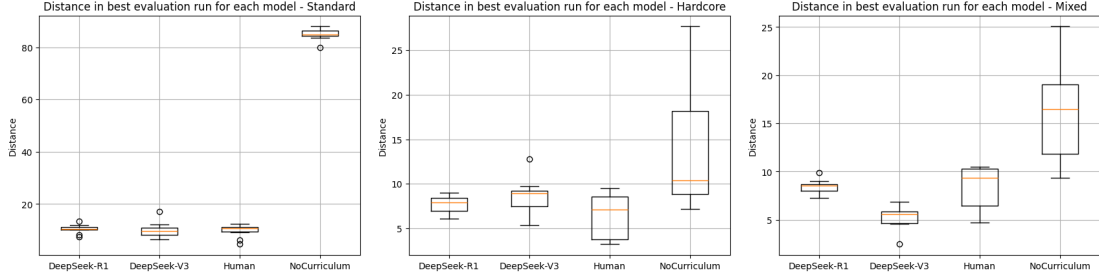


Figure 2: Mean distance traveled in the best evaluation run for each model under Standard, Hardcore, and Mixed environments.

stant is calculated from previous episodes. As this is functionally a trained part of the model, it was loaded in during evaluation as well (without it, any model trained with PPO using CleanRL effectively takes random actions during evaluation, even with good performance during training).

5 Results and Discussion

5.1 Model evaluation

For each curriculum setup, we present the best of three runs. The results are reported in the three environments - Standard, Hardcore, and Mixed – and are summarized in Table 1 and shown in Figure 2.

	Standard	Hardcore	Mixed
R1	10.40 ± 1.59	7.84 ± 0.73	8.46 ± 0.66
V3	10.05 ± 2.90	8.53 ± 2.03	5.22 ± 1.15
Human	9.80 ± 2.37	7.69 ± 2.20	8.37 ± 2.15
NoCurr	85.07 ± 2.20	14.03 ± 7.51	15.91 ± 4.81

Table 1: Average distance traveled per setting in the best run (\pm std)

The No Curriculum baseline, which uses BipedalWalker’s optimized reward function without any curriculum, achieves the highest performance across all environments. Notably, in the Standard setting, PPO reaches an average distance that is eight times greater than the other methods, as illustrated in the box plot. This highlights the strength of well-tuned reward functions compared to the curriculum design.

While LLM-generated curricula do not outperform the PPO baseline, they are competitive with the human-designed curriculum, achieving higher mean distances in two out of three settings - Standard and Hardcore – highlighting that the LLMs are more effective in more standard environments, but performance can decline in high-variance scenar-

ios like Mixed, where human intuition may provide better task decomposition.

Overall, performance tends to be lower in the Hardcore and Mixed environments across all methods. Agents rarely learn to skip obstacles, and only the No Curriculum baseline occasionally learns to handle stairs. According to the summary metrics in the table, the average distances reached in the best runs are generally comparable, with most results falling within each other’s standard deviations. Notably, the DeepSeek-V3 model exhibits greater variability, which allows it to reach higher maximum distances on occasions. On the other hand, the DeepSeek-R1 model even though employed chain-of-thought processes, has a lower variability and maintains modest performance.

The agent does not seem to be explicitly benefiting from a structured curricula in mastering the standard and hardcore environments, however we discuss the potential to isolate which aspect of curricula design may be impacting performance in further experiments (e.g., environmental controls, stage design choices against target task).

Importantly, these curricula were generated with zero-shot prompt and no iterative refinement. Prior work such as Eureka and Revolve demonstrates that incorporating evolutionary strategies – like reward rewriting, or performance-based selection – can significantly boost the performance.

5.2 Training process analysis

The plots in Figure 3 show the learning trajectories of the best-performing runs across all training conditions, displayed to investigate how agents progressed over time with different curricula. The NoCurriculum baseline consistently achieved the highest final distances and showed a smooth and steady upward trend across all environments, confirming its position as a strong benchmark. In contrast, LLM-generated curricula displayed more var-

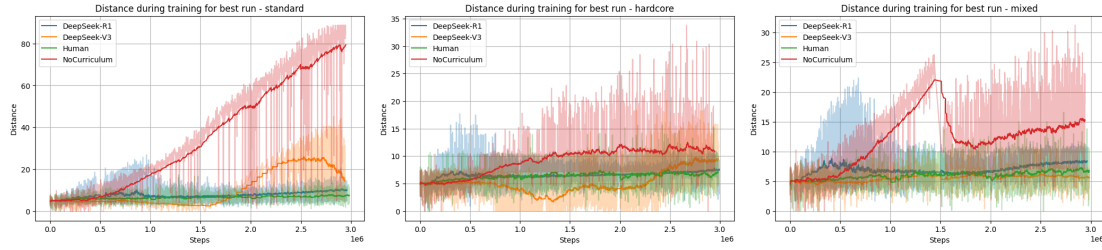


Figure 3: Learning trajectories of the best-performing run for each model under Standard, Hardcore, and Mixed environments.

ied learning behavior. With some curricula generated by Deepseek-V3, performance improved in later stages whereas in others the agent’s survival rate decreased. This suggests the reward functions in some stages may deviate from the overall target task and therefore the agent learns adversarial behavior. DeepSeek-R1, on the other hand, consistently across curricula showed rapid early improvement — especially in the Mixed setup — but performance plateaus early, suggesting potential lack of robustness with long-term reward structure or progression pacing.

Finally, the human-designed curriculum produces stable but modest learning, with few peaks or collapses, likely due to its conservative structure, and is often outperformed by LLM-generated curricula in environments where adaptability is key, such as the Standard and Mixed setups.

Interestingly, while most policies improve from Hardcore to Mixed, V3 appears to perform worse. This is not due to failure in learning, but rather a result of its curriculum inducing a policy that penalizes obstacle detection—leading the agent to back away from obstacles or remain still, thus limiting distance traveled.

5.3 Insights from learning graphs: Potential Pitfalls in the LLM generated curricula

Closer inspection of individual training runs reveals that not all curriculum stages are equally effective, and in some cases, later stages may even introduce counterproductive behavior. This is clearly illustrated in Run 2 of DeepSeek-V3’s Standard setup (see Figure 4) the agents perform well during stage 2, consistently achieving distances over 20 units. However, after transitioning to stage 3 at around 1.5 million steps, it adopts behavior that inhibits its movement causing performance to sharply drop and remain suppressed. This regression could be driven by several factors: one reason could be linked to the reward function generated

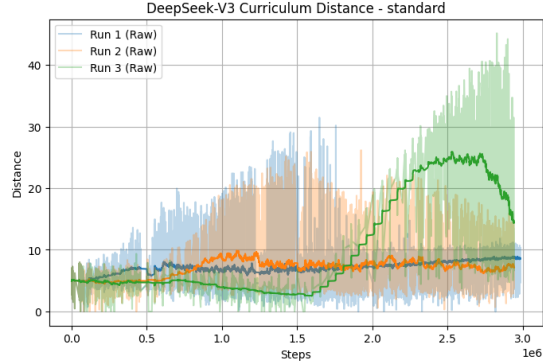


Figure 4: Learning dynamics of DeepSeek-V3 in the Standard environment.

by DeepSeek-V3 for stage 3 being incompatible with the overall objective and emphasizing irrelevant behaviors. Alternatively, the subtask design or sequencing logic might be flawed, disrupting learning progression rather than reinforcing it.

5.4 Comparison of generated curricula

The curricula generated by DeepSeek-R1 generally consist of more complex reward functions. This can be seen in greater considerations for stability and taking lidar readings into account more often for obstacle detection. However, it also seems to display a tendency to “overthink” where it gets stuck in a reasoning loop and hits the max token limit before being able to output any text. This overthinking capacity can be clearly seen in the standard environment runs, where R1 often attempts to incorporate object detection and avoidance into training despite being told multiple times that the environment has no obstacles. R1 would also occasionally fail to follow instructions such as outputting all reward functions at once, or only outputting Python code (an issue that was never encountered with V3).

5.5 Visual comparison of policies

The policies generated by DeepSeek-R1 are generally only able to stand for roughly a second, then fall and terminate. This is true across all environments, including standard. In standard, although R1 travelled a slightly further distance compared to V3 (on average), it was only able to take two steps before terminating in most episodes. V3 on the other hand often learnt a policy where it would perform the splits and shuffle along, leading to longer episodes but slower distance travelled. This type of policy for V3 was common across all environments. This also explains the lower distance score compared to R1 for the mixed environment, as the V3 curriculum generated a policy where the agent would stand still and attempt to avoid obstacles by moving backwards. The R1 policy in comparison simply falls forward on initialisation, indicating that it was not able to meaningfully learn. A “split-shuffle” policy was often reached by the human curriculum across all environments as well. This explains its comparable performance to V3 in hardcore. Of the obstacles in hardcore, only stairs were able to be solved by the no-curriculum model (up and down). Boxes and pitfalls remained insurmountable in both hardcore and mixed, although there were instances in mixed where the agent was able to pass a pitfall only to fall back into it. The human-curriculum models faced the same issue with boxes and pitfalls, and were able to go up one or two stairs, but often got stuck. Both LLMs generated curricula where the agent avoided obstacles, likely due to over-reliance on lidar readings (adding negative reward for the agent detecting obstacles).

6 Conclusion and Further Work

6.1 Conclusion

This study evaluated the zero-shot ability of DeepSeek-V3 and DeepSeek-R1 to generate curricula for reinforcement learning in the BipedalWalker environment. While both models successfully produced executable curricula that led to meaningful agent learning, their performance varied across settings. DeepSeek-V3 demonstrated the strongest results among the LLMs tested, slightly outperforming a human-designed curriculum in two out of three environments. These results highlight the potential of LLMs to structure effective training strategies with minimal guidance.

However, closer analysis of training trajectories and policy behaviors revealed some limita-

tions. In some cases, LLM-generated curricula produced regressions in later stages, with agents adopting suboptimal or adversarial behaviors probably due to misaligned reward functions or flawed task sequencing. DeepSeek-R1, while capable of producing more sophisticated reward structures, sometimes struggled with overcomplex reasoning and V3 occasionally repeated reward logic across stages despite claiming progressive learning objectives. These discrepancies underline the sensitivity of curriculum quality to both reward design and stage logic, especially in zero-shot settings.

Additionally, the ethos of curricula learning is to gradually expose the agent to more difficult tasks. However, the inability to fully control the environment during training (e.g., train on a particular obstacle at a time) limited the potential for both human and LLM-generated curricula to adopt this modular learning process.

Overall, our findings show that while LLMs generate sensible curricula in zero-shot settings, without refinement via human feedback or EA-driven iterations, the agent learns suboptimal behavior and cannot conquer the environment.

6.2 Further Work

There are several promising directions to extend this work. First, benchmarking a broader range of LLMs, including models with stronger domain-specific training, could help isolate whether performance is limited by reasoning ability or reward design proficiency. A promising direction is to use multiple LLMs collaboratively: for instance, using a reasoning-oriented model to generate a natural language curriculum structure and a code-oriented model to translate each stage into executable reward functions.

Second, adaptive curriculum generation — where an LLM proposes one stage at a time, evaluates agent performance after each stage, and modifies subsequent stages accordingly — could increase alignment between curriculum progression and learning dynamics. This strategy introduces feedback into the curriculum pipeline without requiring full evolutionary refinement.

Additionally, including a zero-shot no-curriculum reward function from each LLM would help disentangle whether poor performance derives from flawed reward function design or improper subtask sequencing —e.g., if the LLM-generated curriculum underperforms the

zero-shot no-curriculum reward, the issue likely lies in the structure of the curriculum itself. If the reverse is true, but both fall short of a human or PPO baseline, then reward design may be the limiting factor, even if the task sequencing is effective.

Complementary analysis could also explore curriculum stage reordering, evaluating each permutation to assess whether the original stage sequence was optimal for learning progression.

Finally, expanding experiments to different tasks and environments, particularly those requiring broader generalization or long-horizon planning, would provide a more comprehensive view of LLMs' capacity for autonomous curriculum design in reinforcement learning.

References

- J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Omur Aydogmus and Musa Yilmaz. 2024. [Comparative analysis of reinforcement learning algorithms for bipedal robot locomotion](#). *IEEE Access*, 12:7490–7499.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.
- A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR.
- Mark Chen, Jerry Tworek, Heewoo Jun, and Qiming et al. Yuan. 2021. [Evaluating large language models trained on code](#).
- DeepSeek-AI and Daya Guo et al. 2025. [Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning](#).
- J L Elman. Learning and development in neural networks: the importance of starting small. *Cognition*, 48(1).
- Sebastien Forestier, Remy Portelas, Yoan Mollard, and Pierre-Yves Oudeyer. 2017. [Intrinsically motivated goal exploration processes with automatic curriculum learning](#).
- Farama Foundation. 2024. Gymnasium documentation: Bipedalwalker. https://gymnasium.farama.org/environments/box2d/bipedal_walker/. Accessed: 2025-04-02.
- Rishi Hazra, Alkis Sygkounas, Andreas Persson, Amy Loutfi, and Pedro Zuidberg Dos Martires. 2024. [Revolv: Reward evolution with large language models using human feedback](#).
- Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. 2022. [Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms](#). *Journal of Machine Learning Research*, 23(274):1–18.
- M. Kwon and S. Michael. 2023. Reward design with language models. In *International Conference on Learning Representations (ICLR)*.
- Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. 2023. [Reward design with language models](#).
- Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. 2022. [Evolution through large models](#).
- H. Li, X. Yang, Z. Wang, X. Zhu, J. Zhou, Y. Qiao, X. Wang, H. Li, L. Lu, and J. Dai. 2024. Auto mc-reward: Automated dense reward design with large language models for minecraft. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16426–16435.
- W. Liang, S. Wang, H.-J. Wang, Y. J. Ma, O. Bastani, and D. Jayaraman. 2024. Environment curriculum generation via large language models. In *8th Annual Conference on Robot Learning*.
- Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. [The flan collection: Designing data and methods for effective instruction tuning](#).
- Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. [Eureka: Human-level reward design via coding large language models](#).
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. [Self-refine: Iterative refinement with self-feedback](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 46534–46594. Curran Associates, Inc.
- Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. 2020. [Curriculum learning for reinforcement learning domains: A framework and survey](#).
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023a. [Instruction tuning with gpt-4](#).
- Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023b. [Instruction tuning with GPT-4](#).
- Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. 2020. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*.

- Eduardo Pignatelli, Johan Ferret, Tim Rockäschel, Edward Grefenstette, Davide Paglieri, Samuel Coward, and Laura Toni. 2024. [Assessing the zero-shot capabilities of llms for action evaluation in rl](#).
- M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-B. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.
- N. Rudin, D. Hoeller, P. Reist, and M. Hutter. 2022. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR.
- Kanghyun Ryu, Qiayuan Liao, Zhongyu Li, Koushil Sreenath, and Negar Mehr. 2024. [Curriculum: Automatic task curricula design for learning complex robot skills using large language models](#).
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. [Proximal policy optimization algorithms](#). *arXiv preprint arXiv:1707.06347*.
- David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dhharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. [Mastering chess and shogi by self-play with a general reinforcement learning algorithm](#).
- I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A.” Garg. Progprompt: Generating situated robot task plans using large language models.
- J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma. 2023. Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics. *arXiv preprint arXiv:2309.06687*.
- Z. Tang, D. Kim, and S. Ha. 2021. Learning agile motor skills on quadrupedal robots using curriculum learning. In *International Conference on Robot Intelligence Technology and Applications*, volume 3.
- H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Roziere, N. Goyal, E. Hambro, F. Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. [Voyager: An open-ended embodied agent with large language models](#).
- H.-C. Wang, S.-C. Huang, P.-J. Huang, K.-L. Wang, Y.-C. Teng, Y.-T. Ko, D. Jeon, and I.-C. Wu. 2023b. Curriculum reinforcement learning from avoiding collisions to navigating among movable obstacles in diverse environments. *IEEE Robotics and Automation Letters*, 8(5):2740–2747.
- L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang. 2024. Gensim: Generating robotic simulation tasks via large language models. In *The Twelfth International Conference on Learning Representations*.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. 2024. [Text2reward: Reward shaping with language models for reinforcement learning](#). In *The Twelfth International Conference on Learning Representations*.
- Michael Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. 2023. [Using large language models for hyperparameter optimization](#). In *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

A Appendix

Below you can find details on the model architecture, the mean distance traveled during training for the best LLM runs, and the distance traveled during training for all LLM runs.

Layer	Type	Input Shape	Output Shape	Activation
Critic	Linear	obs_dim	64	Tanh
	Linear	64	64	Tanh
	Linear	64	1	None
Actor (Mean)	Linear	obs_dim	64	Tanh
	Linear	64	64	Tanh
	Linear	64	action_dim	None
Actor (Log Std)	Parameter	1	action_dim	None

Table 2: Neural network architecture of the agent. obs_dim refers to the flattened size of the observation space, and action_dim refers to the size of the action space. The actor network outputs a normal distribution from which actions are sampled. The critic network outputs a single scalar value representing the state value estimate.

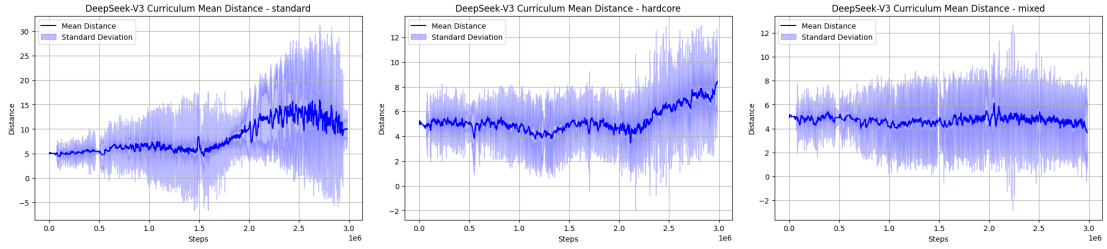


Figure 5: DeepSeek-V3: Mean Distance Traveled across 3 environment set-ups

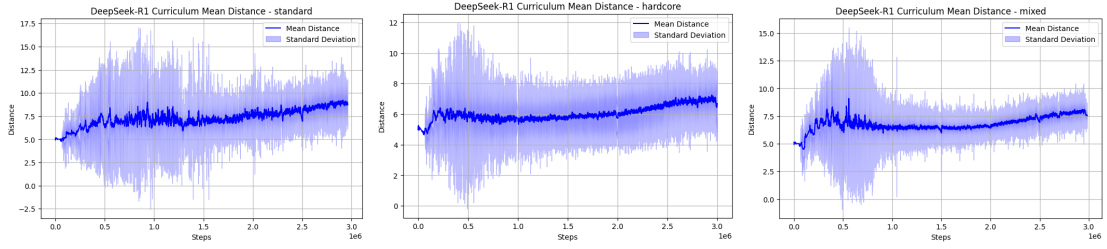


Figure 6: DeepSeek-R1: Mean Distance Traveled across 3 environment set-ups

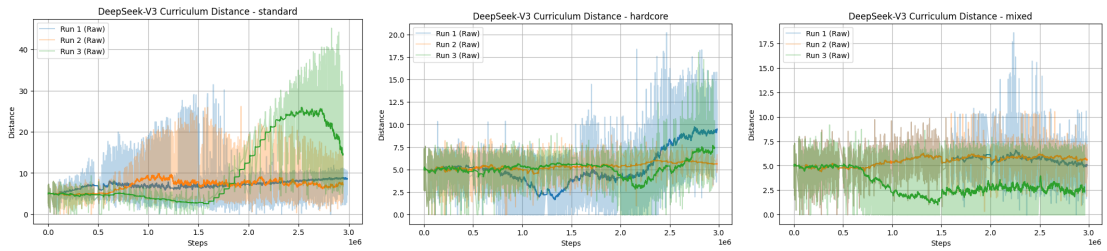


Figure 7: DeepSeek-V3: Distance Traveled by run across 3 environment set-ups with varying curricula

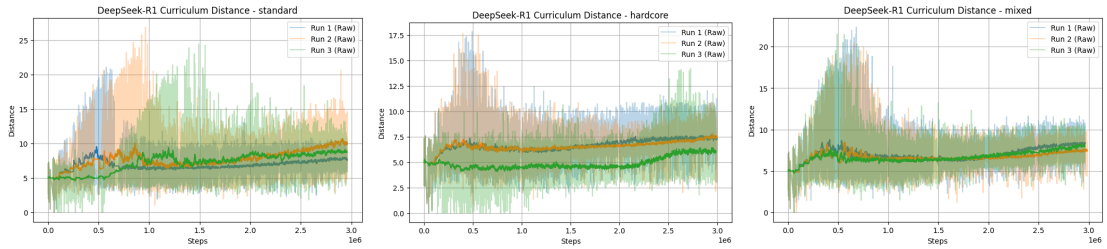


Figure 8: DeepSeek-R1: Distance Traveled by run across 3 environment set-ups with varying curricula