Odyssey Final Report - RLVR For Reasoning LLMs

BEH Chuen Yang

Abstract

This final report summarizes the work done throughout the past weeks on Reinforcement Learning (RL), Large Language Models (LLMs), and RL with Verifiable Rewards (RLVR) in LLMs, culminating in a report detailing how RLVR can be used to train LLMs to perform reasoning tasks. This report presents most of the key results and findings from all weeks of the project, and it is hoped that the report serves as a good roadmap for understanding RLVR's potential to elicit reasoning capabilities in LLMs.

1 Introduction

Large Language Models (LLMs) have long been demonstrated capable of performing a staggering variety of tasks in the Natural Language Processing (NLP) domain, ranging from text generation, sentiment analysis, to machine translation, question answering (Brown et al. (2020)), and in recent years, even complex reasoning tasks (Lambert et al. (2025); Shao et al. (2024); DeepSeek-AI et al. (2025)).

By gaining deeper insights and practical experience in training LLMs with RLVR, one can become better equipped to understand why LLMs are capable of reasoning, how they fall short, and to what extent RLVR can be used to improve their reasoning capabilities.

2 Background

This section informally summarizes the key concepts for understanding RLVR. More details can be found in the Appendices (where applicable) as well as in previous weeks' works.

2.1 LLMs

As their name partly suggests, LLMs are a class of neural networks which attempt to learn the statistical distribution of natural language (Zhao et al. (2025); Karpathy (2025)). Especially in the modern context, LLMs are **by and large** [1] characterized by the following properties (Beh (2025e)):

- LLMs model language *causally* (Jurafsky & Martin (2024); Karpathy (2025)).
- LLMs adopt *some variant of the Transformer architecture* (Vaswani et al. (2017)).
- LLMs require a lot of data (often in the tens of terabytes (Karpathy (2025))) to train.
- LLMs typically have *billions or even trillions* of parameters.
- Modern LLMs typically have *diverse capabilities*, (Brown et al. (2020)) such as generating coherent text, following instructions, and even engaging in conversations.

2.1.1 *How are LLMs trained?*

Per Karpathy (2025), LLMs typically undergo multiple stages of training, developing different capabilities at each stage.

Firstly, during *pre-training*, LLMs are trained on a large corpus of text data to learn general language patterns and structures by optimizing the objective in Equation (2). This stage is

[1] Exceptions exist and are mentioned in Beh (2025e). They are omitted here for brevity.

typically *self-supervised* in that the corpus provides both the conditioning context and the target output, eliding the need for human-curated labels. [2] Next, some LLMs (Ouyang et al. (2022)) undergo *supervised fine-tuning* on more specialized corpora to adapt their capabilities to specific tasks or domains (Radford et al. (2019); Brown et al. (2020)). While the objective is identical to Equation (2) (Beh (2025e)), the corpus *consists of human-annotated examples* and the objective *does not consider the log-likelihoods of the prompt tokens* (since strictly speaking, the prompt tokens are not part of the target output).

Finally, most modern LLMs (Lambert et al. (2025); DeepSeek-AI et al. (2025)) undergo *reinforcement learning post-training*, such as RL with Human Feedback (RLHF) (Christiano et al. (2017); Ouyang et al. (2022)) or RLVR (Lambert et al. (2025); Shao et al. (2024); DeepSeek-AI et al. (2025)) to refine their output quality and alignment with human preferences. We defer discussion of RLVR to Section 3, as it is the main focus of this report.

2.2 RL

In the general case, RL is a machine learning paradigm where an agent learns to take a sequence of *actions* in an *environment* so as to to maximize cumulative reward (Sutton & Barto (2018)).

2.2.1 *RL Problems as Markov Decision Processes (MDPs)*

Most commonly, RL problems are formulated as MDPs by *assuming the Markov property*, where environment dynamics only depend on the current state and action (Achiam (2018); Levine et al. (2023); Sutton & Barto (2018)) [3] This *Markov assumption* is **key** to enabling the theoretical performance guarantees of many RL algorithms (Sutton & Barto (2018)).

Typically, an agent interacts with the environment via a *parameterized, stochastic* policy to collect experiences known as *trajectories* or *episodes*. The agent then uses these trajectories to update its policy parameters in a way that maximizes the expected cumulative reward (Sutton & Barto (2018); Achiam (2018); Levine et al. (2023)).

2.2.2 *Policy Gradient Methods*

Where RLVR is concerned, we are primarily interested in *policy gradient methods* (Beh (2025c); Achiam (2018); Weng (2018)), whose direct optimization on the RL objective in Equation (3) is **uses the observation that the expected cumulative reward is differentiable with respect to the policy parameters** θ (Achiam (2018); Levine et al. (2023); Weng (2018)).

This gives rise to a simple gradient ascent algorithm (Beh (2025c)) which a wide variety of algorithms like REINFORCE (Williams (1992)), Proximal Policy Optimization (PPO) (Schulman et al. (2017)), and Group Relative Policy Optimization (GRPO) (Shao et al. (2024)) are ultimately based on, the details of which we relegate to Algorithm 1 in Appendix C.

2.2.3 *Special Mention: GRPO*

A significant algorithm distinguished for its use in RLVR is the Group Relative Policy Optimization (GRPO) algorithm (Shao et al. (2024)).[4] It is best described as a *computationally-motivated variant* of PPO (Schulman et al. (2017)) without a neural network baseline, which allows it to optimize LLMs on a tight memory budget, and has been shown to be effective in training LLMs with RLVR (Shao et al. (2024); DeepSeek-AI et al. (2025)).

[2]See Appendix B.1 and (Beh (2025f)) for an experiment on pre-training a small LLM on the TinyShakespeare corpus (Karpathy (2015a;b)).

[3]A formal treatment of the following paragraph is given in Appendix A.2.

[4]A summary of the GRPO Objective may be found in Appendix A.3.

3 RLVR

At the intersection of RL and LLMs lies RLVR, which is characterized by its use of RL to steer LLMs' outputs towards *verifiably correct* ones, according to task-specific criteria. (Lambert et al. (2025); Shao et al. (2024); DeepSeek-AI et al. (2025)).

This method is not only very simple to implement and able to extend the frontier of LLM capabilities (Lambert et al. (2025); DeepSeek-AI et al. (2025)), but it also escapes many of the usual pitfalls associated with RL on LLMs. For example, models trained on RLHF (Ouyang et al. (2022)) can provide *incorrect, but convincing* answers to questions, (Wen et al. (2024)), or learn mismatched preferences due to biases mentioned in Pitis (2023).

Formally speaking, RLVR is typically formulated as a contextual bandit problem (Sutton & Barto (2018); Langford & Zhang (2008)) (as with other RL methods on LLMs), where each model output $(o_1, \ldots, o_n)$ denotes a *single action*. A sample objective for RLVR may be obtained by considering Section A.4.

4 Experiments

Equipped with the requisite background knowledge on RLVR, we can now proceed to explore how LLMs can be effectively trained to perform reasoning tasks with this technique.

While this section is a partial summary of the work done in Beh (2025b), it also contains new experiments and findings contributing to our knowledge of reward function design in RLVR.

As in Beh (2025b), we will be testing small, base language models on the GSM8K dataset (Cobbe et al. (2021)) and evaluating their Pass@1 performance on a small subset of holdouts. **However, we will also be varying the reward function in order to *also minimize* the length of models' responses, in addition to maximizing the correctness of their answers.**

4.1 GSM8K Dataset

GSM8K (Cobbe et al. (2021)) is a grade school level math problem dataset consisting of 8,792 word problems. All demand a single integer as the answer, and all were deliberately designed to be solvable by elementary school students.

Due to the natural language format, this task is difficult to solve algorithmically. Instead, it concurrently assesses models' ability to parse diverse problem statements, reason about numbers, and perform arithmetic operations accurately (Cobbe et al. (2021)).

4.2 Experimental Configurations

For all experiments, we relied on an adapted version of the nano-aha-moment repository (Kazemnejad et al. (2025)), which provides a faithful single-file implementation of the GRPO algorithm (Shao et al. (2024)) and a simple training pipeline for arbitrary LLMs on arbitrary datasets (as long as they are supported by HuggingFace Transformers (Wolf et al. (2020))) (Beh (2025b)).

Training was alternately conducted on a single NVIDIA 4090 GPU with 48GB of VRAM, and a single NVIDIA 3090 GPU with 24GB of VRAM [5]. All runs took a combined ~ 40 hours of wall-clock time to complete. As in Beh (2025b), we only performed one trial for each reward function due to compute restrictions.

4.3 Reward Modelling

Owing to its central role as an independent variable in our experiment, we go into further detail on the design of the reward function in Beh (2025b).

[5]Thanks Chann.

In Beh (2025b), our reward function is a variant of Kazemnejad et al. (2025)'s, and it is a more permissive variant of that described by DeepSeek-AI et al. (2025):

$$\begin{aligned} R(s,a) &= C(s,a) + F(s,a) \\ &= \begin{cases} 1 & \text{if } a \text{ contains the correct answer} \\ 0 & \text{otherwise} \end{cases} \\ &+ \begin{cases} 1 & \text{if } a \text{ contains thinking, } \backslash n\text{, and a single integer answer.} \\ 0.5 & \text{if } a \text{ contains thinking, } \backslash n\text{, and any answer.} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

We speculate the additional half-reward clause encourages the model to produce valid answers even if they do not obey the strict response format (Beh (2025b)).

4.4 Hyperparameters

The numerical hyperparameters used in our experiments were identical to those used in Beh (2025b). For more details, please refer to Table 2.

In attempting to minimize the length of the model's responses, we also considered the following reward functions (the default reward function from Beh (2025b) is also included for comparison):

Reward Function	**Description**	**Function Name**
Control (Beh (2025b))	$r(s,a) = R(s,a)$ (Equation 1)	`noop_decay`
Linear Penalty	$$r(s,a) = \left(1 - 0.5clip\left(\frac{len(a)}{L}, 1\right)\right) R(s,a)$$	`linear_length_decay`
Quadratic Penalty	$$r(s,a) = \left(1 - 0.5clip\left(\left(\frac{len(a)}{L}\right)^2, 1\right)\right) R(s,a)$$	`quadratic_length_decay`
Batch Exponential Moving Average Penalty	$$r(s,a) = \beta R(s,a)$$ $$\beta = \begin{cases} 1.2 & \text{if } a < \texttt{avg\_response\_length} - 10 \\ 0.8 & \text{if } a > \texttt{avg\_response\_length} + 10 \\ 1.0 & \text{otherwise} \end{cases}$$	`batch_exp_avg_decay`
ShorterBetter (Yi et al. (2025))	$$r(s,a) = R(s,a) - 0.0005\lvert len(a) - l_{SOL}\rvert$$ $$l_{SOL} = \begin{cases} min(len(a_1), \ldots, len(a_n)) & \text{if any } a_i \text{ is correct} \\ \frac{1}{n}\sum_{i=1}^{n} len(a_i) & \text{otherwise} \end{cases}$$	None (See `nano_r1_gsm8k_shorterbetter.py`)

Table 1: Reward Functions used in our experiments.

5 Results

5.1 Pass@1 Performance

6 Discussion

References

Joshua Achiam. Spinning Up in Deep Reinforcement Learning. 2018. URL https://spinningup.openai.com/en/latest/.

Chuen Yang Beh. Week 1: Reinforcement learning preliminaries, 2025a. URL ../wk1/wk1.pdf.

Chuen Yang Beh. Week 10: Deep dive into rlvr with nano-aha-moment, 2025b. URL ../wk10/wk10.pdf.

Chuen Yang Beh. Week 2: Foundations of policy gradient methods, 2025c. URL ../wk2/wk2.pdf.

Chuen Yang Beh. Weeks 3 & 4: Implementing an rl training pipeline, 2025d. URL ../wk3/wk3.pdf.

Chuen Yang Beh. Week 5: Introducing large language models to a general audience, 2025e. URL ../wk5/wk5.pdf.

Chuen Yang Beh. Week 8: Pre-training with nanogpt, 2025f. URL ../wk8/wk8.pdf.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Paul Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences, 2017. URL https://arxiv.org/abs/1706.03741.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Daniel Jurafsky and James H. Martin. Speech and language processing, 2024. URL https://web.stanford.edu/~jurafsky/slp3/. Draft of the 3rd edition, available online.

Andrej Karpathy. Tiny shakespeare, 2015a. URL https://raw.githubusercontent.com/karpathy/char-rnn/master/data/tinyshakespeare/input.txt.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, 2015b. URL https://karpathy.github.io/2015/05/21/rnn-effectiveness/#Shakespeare.

Andrej Karpathy. nanogpt, 2022. URL https://github.com/karpathy/nanoGPT.

Andrej Karpathy. Deep dive into llms like chatgpt, 2025. URL https://www.youtube.com/watch?v=7xTGNNLPyMI.

Amirhossein Kazemnejad, Milad Aghajohari, Alessandro Sordoni, Aaron Courville, and Siva Reddy. Nano aha! moment: Single file ”rl for llm” library. https://github.com/McGill-NLP/nano-aha-moment, 2025. GitHub repository.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. 2008.

Sergey Levine, Kyle Stachowicz, Vivek Myers, Joey Hong, and Kevin Black, 2023. URL https://rail.eecs.berkeley.edu/deeprlcourse/.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Silviu Pitis. Failure modes of learning reward models for LLMs and other sequence models. In *ICML 2023 Workshop The Many Facets of Preference-Based Learning*, 2023. URL https://openreview.net/forum?id=NjOoxFRZA4.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2019. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

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, 2024. URL https://arxiv.org/abs/2402.03300.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. Gymnasium: A standard interface for reinforcement learning environments, 2024. URL https://arxiv.org/abs/2407.17032.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, Illia Polosukhin, Samy Bengio, and Yiming Yang. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

Jiaxin Wen, Ruiqi Zhong, Akbir Khan, Ethan Perez, Jacob Steinhardt, Minlie Huang, Samuel R. Bowman, He He, and Shi Feng. Language models learn to mislead humans via rlhf, 2024. URL https://arxiv.org/abs/2409.12822.

L. Weng. Policy gradient algorithms, 2018. URL https://lilianweng.github.io/posts/2018-04-08-policy-gradient/.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992. URL https://people.cs.umass.edu/~barto/courses/cs687/williams92simple.pdf.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Jingyang Yi, Jiazheng Wang, and Sida Li. Shorterbetter: Guiding reasoning models to find optimal inference length for efficient reasoning, 2025. URL https://arxiv.org/abs/2504.21370.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025. URL https://arxiv.org/abs/2303.18223.

A Objective Functions

A.1 LLM Pre-training Objective

Where θ^* are the model parameters and N is the model's context length, an optimal language model $LM_{(\theta^*,N)}$ maximizes the likelihood of the next token given all previous tokens, across all training examples $\left\{ \{x_{j+i}\}_{i=0}^{N-1} \right\}_{j=1}^{M-N}$ in a corpus of size M:

$$\theta^* = \arg\min_{\theta} \sum_{j=1}^{M-N} \left[\sum_{i=0}^{N-1} - \log P_\theta(x_{j+i} | x_j, \cdots, x_{<(j+i)}) \right] \quad (2)$$

A.2 General RL Objective

Formally speaking, an MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \tau, r, \gamma)$, where $\mathcal{S}$ is the set of *states* in the environment, $\mathcal{A}$ is the set of *actions* the agent can take, $\tau : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the *transition function*, $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ is the *reward function*, and $\gamma \in [0, 1]$ is (an optional) *discount factor*.

Typically, an agent interacts with the environment via a *parameterized, stochastic* policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, in an *episodic* manner, where $s_0 \sim \rho_0$ (the initial state distribution), and $\forall_{0 \le k < n} (a_k \sim \pi_\theta(s_k), s_{k+1} \sim \tau(s_k, a_k))$ (Beh (2025a)). The agent thus induces a sequence of interactions $T = (s_0, a_0, s_1, a_1, \ldots, s_n)$ (a *trajectory*), which should be regarded analogously to one of many games of Tic-Tac-Toe, or a single playthrough of a video game.

Simply put, the optimal policy π_{θ^*} is the one that maximizes the *expected cumulative reward* (Beh (2025c)):

$$\pi_{\theta^*} = \arg\max_{\pi_\theta} \mathop{\mathbb{E}}_{T \sim \pi_\theta} \left[\sum_{t=0}^{n} \gamma^t r(s_t, a_t) \right] \quad (3)$$

A.3 GRPO Objective

Consider a policy network π_θ and critic network V_ϕ in a single-state, single-action armed bandit environment (Sutton & Barto (2018); Langford & Zhang (2008)) with state s and action $a = (o_1, \ldots, o_n)$, where o_k is the k-th output token of the language model. Then the GRPO objective is defined as follows (Shao et al. (2024); Beh (2025b)):

$$\mathcal{J}(\theta) \quad = \mathbb{E}_{(s, o_1, \ldots, o_n) \sim \pi_\theta} \left[\frac{1}{n} \sum_{k=1}^{n} \min \left\{ \frac{\pi_\theta(o_k | s, o_{<k})}{\pi_{\theta_{old}}(o_k | s, o_{<k})} A(s, a), g(s, a) \right\} \right] - \lambda D_{KL}(\pi_\theta || \pi_{\theta_{old}}) \quad (4)$$

where $g(s, a) = clip \left(\frac{\pi_\theta(o_k | s, o_{<k})}{\pi_{\theta_{old}}(o_k | s, o_{<k})}, 1 - \epsilon, 1 + \epsilon \right) A(s, a)$, and $\pi_{\theta_{old}}$ is the policy network from the previous GRPO step.

A.4 RLVR Objective

GRPO (Shao et al. (2024)) has historically been the algorithm of choice for RLVR (DeepSeek-AI et al. (2025); Shao et al. (2024)), hence an objective function for RLVR can simply be 4.

B Miscellaneous Case Studies

B.0.1 Case Study: REINFORCE

In Beh (2025d), we experimented with a simple RL training pipeline by Levine et al. (2023) to train a simple network on the CartPole environment (Towers et al. (2024)).

B.1 Case Study: nanoGPT

In Beh (2025f), experiments in pre-training a small LLM on the TinyShakespeare corpus (Karpathy (2015a;b)) were conducted with nanoGPT (Karpathy (2022)). While the details of the experiment can be found in Beh (2025f), we summarize the key findings here:

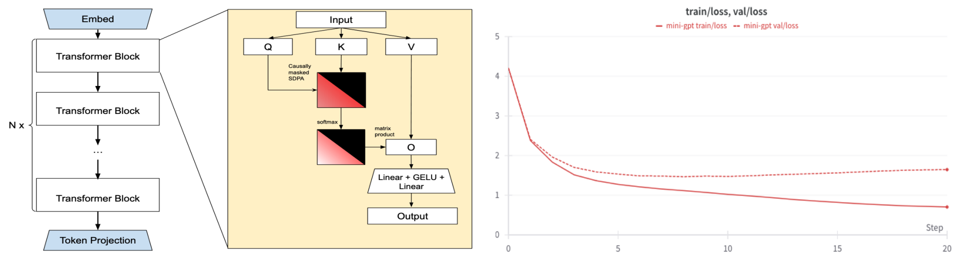

Figure 1: (Left) The architecture of nanoGPT, illustrated. (Right) The training loss curves of the pre-trained nanoGPT model.

- Despite being tokenized on a character level, the model could generate valid whole words with few to no errors.
- The model could generate text that (superficially) resembled the training corpus.
- The model often produced nonsensical text.

C Policy Gradient Algorithm

Algorithm 1 Improved Policy Gradient Algorithm

Input: Policy π_θ, learning rate α, baseline of choice $b(s_t)$
Output: Updated policy π_θ
while not converged **do**
    Sample a batch of trajectories T from the policy π_θ
    for each trajectory T in the batch **do**
        Compute the rewards-to-go R_t for each time step t in the trajectory
        Compute the baseline $b(s_t)$ for each time step t in the trajectory
        Estimate $\nabla_\theta J(\pi_\theta) \approx \frac{1}{m}\left(\sum_{t=0}^{n} R_t - b(s_t)\right)\left(\sum_{t=0}^{n} \nabla_\theta \log(\pi_\theta(a_t|s_t))\right)$
    end for
    $\theta \leftarrow \theta + \alpha \nabla_\theta J(\pi_\theta)$
end while

D Numerical Hyperparameters

As in Beh (2025b), we used the following hyperparameters for all runs of the GSM8K task:

Hyperparameter	GSM8K
Model Family	Qwen2.5 (Qwen et al. (2025))
Model Scale	1.5B
Batch Size	16
Learning Rate	1e-6
Max. Response Length (L)	512
Completions per Task	8
Episodes per GRPO Iteration	16
KL Divergence Coefficient λ	0
Random Seed	42

Table 2: Universal Hyperparameters across our experiments.