VIRAL: Vision-grounded Integration for Reward design And Learning

Valentin Cuzin-Rambaud[a], Emilien Komlenovic[a], Alexandre Faure[a] and Bruno Yun[a,b]

[a]Université Claude Bernard Lyon 1, France
[b]CNRS, Ecole Centrale de Lyon, INSA Lyon, Université Lumière Lyon 2, LIRIS, UMR5205, France

Abstract. The alignment between humans and machines is a critical challenge in artificial intelligence today. Reinforcement learning, which aims to maximize a reward function, is particularly vulnerable to the risks associated with poorly designed reward functions. Recent advancements has shown that Large Language Models (LLMs) for reward generation can outperform human performance in this context. We introduce VIRAL, a pipeline for generating and refining reward functions through the use of multi-modal LLMs. VIRAL autonomously creates and interactively improves reward functions based on a given environment and a goal prompt or annotated image.

The refinement process can incorporate human feedback or be guided by a description generated by a video LLM, which explains the agent's policy in video form. We evaluated VIRAL in five Gymnasium environments, demonstrating that it accelerates the learning of new behaviors while ensuring improved alignment with user intent. The source-code and demo video are available at: https://github.com/VIRAL-UCBL1/VIRAL and https://youtu.be/Hqo82CxVT38.

1 Introduction

Reward shaping [15] is a fundamental challenge in Reinforcement Learning (RL), involving the design of reward functions that efficiently guide an agent towards desired behaviors. A poorly designed reward can lead to unintended behaviors, while a well-crafted one accelerates learning and ensures alignment with human intent. However, designing effective rewards is labor-intensive and requires significant expertise, particularly in complex environments [2, 6, 9, 16].

Early work on automated reward design [3] leveraged natural language processing techniques — such as recurrent neural networks and word embeddings — to construct reward signals, demonstrating promising results in ATARI games. More recently, LLMs have gained traction in RL and robotics due to their versatility in solving diverse problems [7, 18]. Early attempts at using LLMs for reward shaping [8] employed GPT-3 as a binary reward signal, but this approach was limited in scope and applicability. More recent advances [12, 19, 23] have leveraged OpenAI's GPT-4 model to generate code for reward functions, demonstrating competitive performance and improved adaptability across different environments. However, these methods only focus on the text-based inputs (disregarding vision) and their reliance on a closed-source, and computationally expensive LLM for performance, hinders reproducibility and accessibility.

We propose a **V**ision-grounded **I**ntegration for **R**eward design **A**nd **L**earning (VIRAL) framework to design rewards functions from simple users prompts and/or annotated image. VIRAL sets itself from the state-of-the-art [12, 23] in several key ways. First, VIRAL prioritizes the use of open-source, efficient, and lightweight LLMs [5, 4], ensuring greater accessibility, cost efficiency, and transparency. Second, unlike prior methods, VIRAL integrates Large Vision Language Models (LVLMs) [11, 13] to process both text and images, enhancing its ability to interpret user intent more accurately. Third, VIRAL is the first reward shaping approach to incorporate Video-LVLMs [10] for describing the movements of objects within the scene, providing richer context for reward function generation. Finally, instead of relying on direct access to the environment's code (as in EUREKA [12]) or structured abstraction (through Pydantic class definitions as in Text2Reward [23]), VIRAL describes environments solely through its observations, following the Gymnasium [21] documentation. This simplifies implementation for users while ensuring the LLM captures the necessary information for coherent reward generation. Our main contributions are as follows: (1) The VIRAL pipeline to automatically design reward functions using a simple natural language prompt and/or annotated image. (2) A self-refinement process for reward functions, augmented with human feedback or video description from a LVLM. (3) A scalable and modular implementation designed to adapt to various RL problems within the Gymnasium framework, leveraging multiprocessing for faster inference. (4) An evaluation of VIRAL in five Gymnasium environments, showing its various benefits for learning and user intent alignment, using an empirical study.

This paper is organized as follows: Section 2 details the architecture and inner workings of VIRAL. Section 3 presents our evaluation methodology and results. Section 4 provides concluding remarks.

2 The VIRAL Architecture

This section details the VIRAL procedure to generate multiple rewards functions which are rated to find the best agent behaviors.

2.1 The Input Parameters

For its input, VIRAL uses a set of specific elements (see top of Figure 1). It includes (a_1) a textual environment description d_{env}, (a_2) an optional implementation of a success function f_{succ} provided as Python code, and (b) the goal prompt p_g (which can be multi-modal).

To give the prompt p_g, the user has the possibility to choose between a text prompt, an annotated image img, or both of those. The img can include annotations (with arrows, text, areas, etc.). The input d_{env} is extracted from Gymnasium and provides an accurate representation of the environment's observable space to the LLM, adding

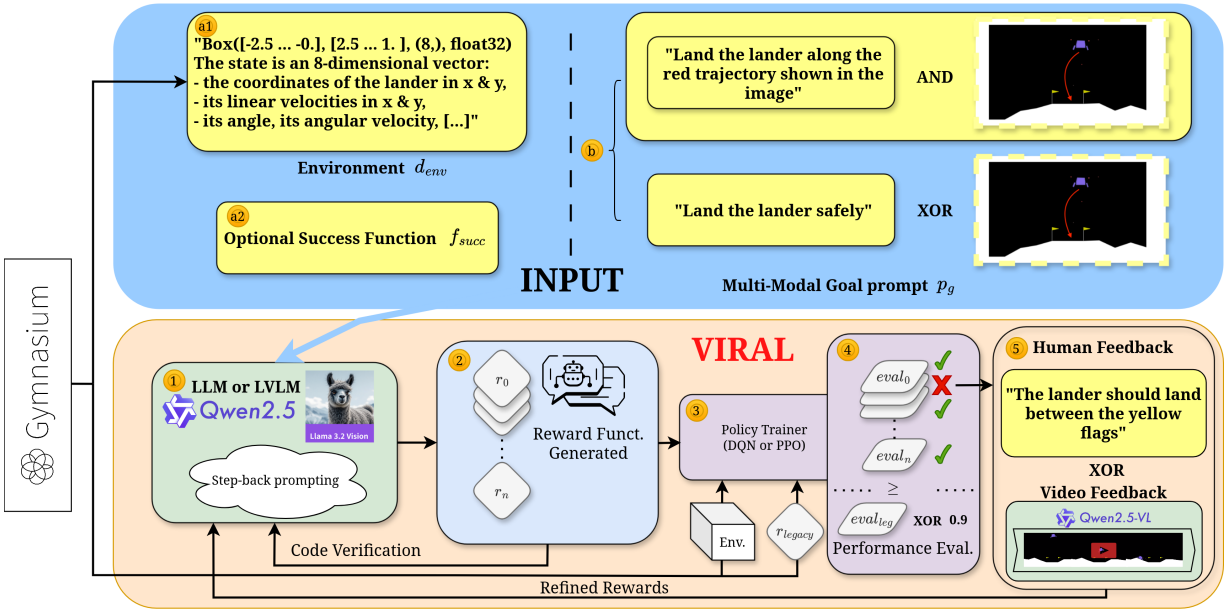

Figure 1. The VIRAL pipeline. Given an input (a textual environment description, an optional success function, and a goal prompt), the system generates a set of reward functions and iteratively refines them.

more depth to the generated reward function. Using text for d_{env} allows for any Gymnasium environment to be seamlessly integrated with VIRAL. The input $f_{succ} : States(env) \rightarrow \{0, 1\}$ is a success function which must be tailored to the goal. This function determines how success is manifested within the specific context of the task, helping the LLM in designing the reward function. Note that while f_{succ} must be related to the goal prompt p_g, returning 1 (success) can mean a partial success. For example, for a textual goal prompt p_g : "Land the lander along the red trajectory shown in the image", f_{succ} may return 1 if the lander managed to land, ignoring the trajectory.

2.2 *The Initial Generation*

For the initial generation, we opted for zero-shot prompting, despite the effectiveness of few-shot prompting [1]. This was motivated by generalization needs and the difficulty for users to provide examples, making zero-shot prompting more practical and user-friendly.

VIRAL implements a collaboration between two LLMs (critic and coder), with specific system prompts for their roles. The former must be a good supervisor and specify steps to help the latter in producing good quality code. Among the strategies employed to enhance zero-shot generation, one particularly effective method is step-back prompting [25], which allows to obtain a broader perspective by first reasoning about a problem at a higher-level before generating a detailed response (see step 1 of Fig. 1). The critic LLM generates the step-back prompt which is given to the coder LLM (see step 2 of Fig. 1). Note that only the critic LLM needs to be multi-modal.

The generated code is checked for syntax and logical issues by using a try-catch process. If errors are caught, they are sent back to the coder LLM via a custom prompt, allowing it to revise its output.

2.3 *Learning a Policy*

The generated reward functions take an observation as parameters, along with two boolean values indicating whether there is a success or failure[1]. Once a reward function is generated, an RL algorithm is used to make the agent search for its policy. We restrict ourselves to Deep Q-Network (DQN) [14] and Proximal Policy Optimization (PPO) [17] and select the most suitable one for each environment (step 3 of Fig. 1). During training, rewards and observations are retrieved, and if the user wishes to, they can implement objective metric functions that take these observations and return a dictionary of useful objective metrics for this environment. During testing we can compare the success rate of our custom reward, with the baseline "legacy" reward function r_{legacy} or with a user-defined threshold, which determines an acceptable success rate (step 4 of Fig. 1).

2.4 *The Refinement Process*

The refining process unfolds as follows (see step 5 of Fig. 1).

The critic LLM analyzes the training results by leveraging collected statistics like state-observations during the run, and an optional user or Video-LVLM feedback. Indeed, to identify potential reasons why the reward function underperformed, a user or Qwen2.5-VL feedback can provide a more precise description of the agent's learned behavior. These observations are passed to the coder LLM, which creates an improved reward function by addressing the identified weaknesses and aligning with the intended objectives

These methods are combined into an iterative approach. In each iteration, a refined reward function is evaluated and compared to its previous version. If it does not pass the evaluation, the refinement process is repeated until the desired performance level is achieved.

3 Empirical evaluation

For our evaluation, we used *Qwen2.5-Coder-32B* [24, 5] as the coder LLM due to its performance being comparable to that of *GPT-4*, making it a reliable choice for this role. For the multi-modal critic LLM, we chose *Llama3.2-Vision-11B* [13], as it is lightweight,

[1] Note that an agent can be in a neutral state, without a success or a failure.

open-source, and well-suited for our framework's requirements. For Video-LVLM, we used *Qwen2.5-VL-7B* [22, 20][2]. The evaluation was performed with a NVIDIA A40 GPU and a 12-cores Intel CPU.

In our experiments, we used a selection of environments from the Gymnasium toolkit to evaluate the performance of our generated reward functions. Namely, the classic environments of *CartPole* and *Lunar Lander* allowed us to test fundamental control and optimization strategies in simple yet challenging scenarios. Next, we selected the *Highway* environment, where a vehicle must navigate a multi-lane road, avoiding collisions and optimizing its speed while adhering to driving rules. Given the increasing relevance of autonomous vehicles, we considered it essential to include this environment in our study as it addresses modern-day challenges in automation and decision-making. Finally, we incorporated two robotics-focused environments, *Hopper* and *Swimmer*, both derived from the MuJoCo physics simulator. These environments played a crucial role in evaluating the efficiency of the generated reward function for robotic locomotion and control systems.

All results are available in CSV format in our GitHub repository.

Better Rewards. We evaluated the efficiency of VIRAL by comparing the behavior of agents trained using our generated reward function (without the refining process) against those trained with the legacy reward function. We instantiated VIRAL with Qwen2.5-coder-32B as the critic/coder LLM (text-only goal prompt).

For the CartPole environment trained with PPO, and the goal prompt *"Create a reward function for the CartPole environment that encourages keeping the pole upright for as long as possible."*, Table 1 shows that the policy learned with our reward function outperforms that of Gymnasium. Additionally, we can see that the cart moves slightly more on average, resulting in a more stable pole overall.

Table 1. Comparison of legacy reward vs. ours (avg. over 10 runs)

Reward function	Cart position diff.	Pole angle diff.	Success rate
gymnasium	0.281202	0.064587	0.58700
ours	0.308173	0.062499	**0.85300**

For the *Highway* environment trained with DQN, and the goal prompt *"Control the ego vehicle to reach a high speed without collision."*, we have better success rates than the legacy reward function 7 times out of 10 (with a median success rate of 0.78).

Semantic Alignment. We conducted a study with 25 annotators to determine whether the learned behavior is aligned with the goal prompt provided (text, image or both)[3]. Each annotator annotated a subset of 120 videos (4 environments and 10 videos per goal prompt modality). On average, each annotator rated 71 videos for a total of 1777 annotated videos. For each video, annotators used 5-point Likert scale (from "Strongly disagree" to "Strongly agree") to answer the following two questions: (1) *I understand the instructions*, and (2) *The instructions are followed*. The human evaluation allowed us to assess the effectiveness of the modality on the semantic alignment.

For the first item, we obtained a mean of 4.89 ± 0.41 and with only 147 videos with an understanding score less than 5/5, highlighting that very few goal prompts were misunderstood, confirming that goal prompts were well-crafted and understandable.

Fig. 2 shows the mean ratings for the second item and for each environment and modality. Overall, although using a visual goal prompt was not as good as a text-only goal prompt, we still obtained a mean of 2.14 ± 1.42 (out of 5) across environments.

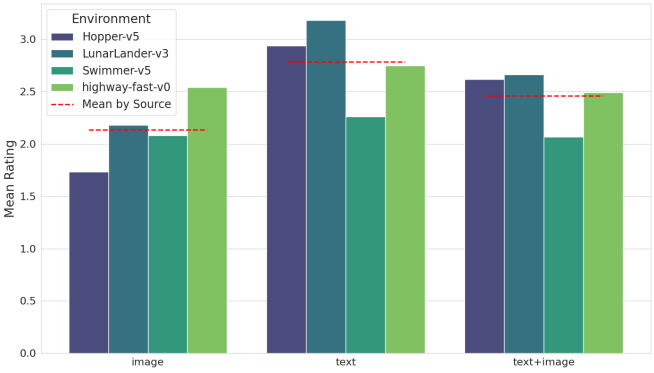

Figure 2. Semantic alignment for different environments and modalities.

However, as VIRAL's main goal is to find the best reward, we investigated which modality led to very good behavior. Table 2 shows that, with the image modality, we discover behaviors that align more than with other modalities on Swimmer and Highway.

Table 2. Maximum selection of average rating per video, comparison by modality.

Environment	Image p_g	Textual p_g	Image+Textual p_g
Hopper-v5	3.41±1.00	**4.92±0.28**	4.83±0.39
LunarLander-v3	3.83±1.26	4.92±0.28	**4.93±0.26**
Swimmer-v5	**4.19±0.75**	4.14±0.95	3.41±1.33
Highway-fast-v0	**4.44±1.29**	4.06±1.03	3.91±1.38

These observations led us to the conclusion that different modalities can help us discover behaviors that align more closely with human intent, despite the number of correct behaviors in average.

Enhanced with Feedback. To show the improvement of a generated reward function brought by the refining process, we can compare its performance to the one of a reward function after a single iteration of feedback obtained by Qwen2.5-VL-7B.

In the LunarLander environment and defining the success criterion as a safe landing between the two yellow poles, we carried out 10 runs, with and without feedback and 100 tests at the end of each training session. We found that the use of the Video-LVLM creates reward functions that are on average 18.33% better based on the success rate.

4 Conclusion

We introduced VIRAL, a pipeline for generating and refining reward functions through the use of multi-modal LLMs. We demonstrated that our approach outperformed legacy reward functions in terms of performance and flexibility. Our results showed that agents were able to learn new behaviors based on simplistic sketches, highlighting the robustness of our approach. Furthermore, the integration of feedback (from a Video-LVLM or a user) enabled agents to better align with the intended objectives, providing a deeper understanding of the desired behaviors.

In future work, we plan to adapt pre-existing policies to learn new behaviors. This approach could pave the way for greater policy generalization and smoother transitions between different sets of complex tasks.

[2] Our framework is model-agnostic and allows the use of any LLM for the coder and critic LLMs and Video-LVLM.

[3] For a full description of the goal prompt used, we refer the reader to our Github repository.

References

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

[3] P. Goyal, S. Niekum, and R. J. Mooney. Using natural language for reward shaping in reinforcement learning. *arXiv preprint arXiv:1903.02020*, 2019.

[4] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[5] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Dang, et al. Qwen2.5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[6] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. Reward learning from human preferences and demonstrations in atari. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8022–8034, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/8cbe9ce23f42628c98f80fa0fac8b19a-Abstract.html.

[7] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K. Lee, Y. Kuang, S. Jesmonth, N. J. Joshi, K. Jeffrey, R. J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and C. K. Fu. Do as I can, not as I say: Grounding language in robotic affordances. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Conference on Robot Learning, CoRL 2022, 14-18 December 2022, Auckland, New Zealand*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2022. URL https://proceedings.mlr.press/v205/ichter23a.html.

[8] M. Kwon, S. M. Xie, K. Bullard, and D. Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023.

[9] K. Lee, L. M. Smith, and P. Abbeel. PEBBLE: feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6152–6163. PMLR, 2021. URL http://proceedings.mlr.press/v139/lee21i.html.

[10] B. Lin, Y. Ye, B. Zhu, J. Cui, M. Ning, P. Jin, and L. Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[11] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[12] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[13] Meta. Llama-3.2-11b-vision-instruct. https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct, 2024.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning, Dec. 2013.

[15] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 278–287. Morgan Kaufmann, 1999.

[16] J. Park, Y. Seo, J. Shin, H. Lee, P. Abbeel, and K. Lee. SURF: semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=TfhfZLQ2EJO.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms, Aug. 2017.

[18] 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. In *IEEE International Conference on Robotics and Automation, ICRA 2023, London, UK, May 29 - June 2, 2023*, pages 11523–11530. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10161317. URL https://doi.org/10.1109/ICRA48891.2023.10161317.

[19] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma. Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics. *arXiv preprint arXiv:2309.06687*, 2023.

[20] Q. Team. Qwen2.5-vl, January 2025. URL https://qwenlm.github.io/blog/qwen2.5-vl/.

[21] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[22] P. Wang, S. Bai, S. Tan, S. Wang, Z. Fan, J. Bai, K. Chen, X. Liu, J. Wang, W. Ge, Y. Fan, K. Dang, M. Du, X. Ren, R. Men, D. Liu, C. Zhou, J. Zhou, and J. Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[23] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[24] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[25] H. S. Zheng, S. Mishra, X. Chen, H.-T. Cheng, E. H. Chi, Q. V. Le, and D. Zhou. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*, 2023.