Verifier-Guided Action Selection For Embodied Agents

Nishad Singhi, Christian Bialas, Snehal Jauhri, Vignesh Prasad,
Georgia Chalvatzaki, Marcus Rohrbach, and Anna Rohrbach

***Abstract*— Building generalist embodied agents capable of solving complex real-world tasks remains a fundamental challenge in AI. Multimodal Large Language Models (MLLMs) have significantly advanced the reasoning capabilities of such agents through strong vision-language knowledge and chain-of-thought (CoT) reasoning, yet remain brittle when faced with challenging out-of-distribution scenarios. To address this, we propose Verifier-Guided Action Selection (VEGAS), a test-time framework designed to improve the robustness of MLLM-based embodied agents through an explicit verification step. At inference time, rather than committing to a single decoded action, VEGAS samples an ensemble of candidate actions and uses a generative verifier to identify the most reliable choice, without modifying the underlying policy. Crucially, we find that using an MLLM off-the-shelf as a verifier yields no improvement, motivating our LLM-driven data synthesis strategy, which automatically constructs a diverse curriculum of failure cases to expose the verifier to a rich distribution of potential errors at training time. Across embodied reasoning benchmarks spanning the Habitat and ALFRED environments, VEGAS consistently improves generalization, achieving up to a 36% relative performance gain over strong CoT baselines on the most challenging multi-object, long-horizon tasks. Code is available at `https://github.com/nishadsinghi/vegas`.**

I. Introduction

A longstanding goal in AI is to create embodied agents that operate autonomously in physical environments to accomplish complex tasks specified through natural language [1], [2], such as navigating to target locations [3] and manipulating everyday objects [4], [5]. Recently, Multimodal Large Language Models (MLLMs), pretrained on Internet-scale vision-language data, have emerged as a promising foundation for building such agents, owing to their strong perceptual and linguistic generalization [6], [7], [8]. While early efforts relied on the zero-shot capabilities of MLLMs [7], [9], [10], finetuning on embodied data—either via supervised learning [11] or reinforcement learning [8], [12], [13]—yields significant improvements. More recently, incorporating Chain-of-Thought (CoT) reasoning has further enhanced decision-making by enabling agents to reason step-by-step before acting [14], [15], [16], [13]. Despite this progress, MLLM-based embodied agents remain brittle in out-of-distribution scenarios and long-horizon settings [6]. For instance, an agent might reliably execute "bring me a banana" but fail when the same goal is phrased as "bring me a yellow curved fruit." Similarly, an agent trained on single-object pick-and-place may fail on a multi-step task such as cleaning an apple and placing it in a cabinet.

We observe that a key factor underlying these failures is that agents cannot recognize mistakes in their reasoning process and correct them at test time. In particular, they commit to a single greedily decoded action at each step with no opportunity for self-correction. In contrast, humans routinely consider multiple candidate actions, mentally evaluate their likely outcomes, and commit only to the most promising one, effectively performing verification before acting. This idea has a direct computational analogue: recent work on scaling test-time compute shows that sampling multiple candidate solutions and selecting the best one via a learned verifier substantially improves LLM performance in domains such as coding and mathematics [17], [18], [19]. However, extending verification to high-level embodied reasoning poses distinct challenges: unlike mathematics or code, embodied agents operate under partial observability and must reason about

The authors are affiliated with TU Darmstadt, Germany and Hessian.AI.

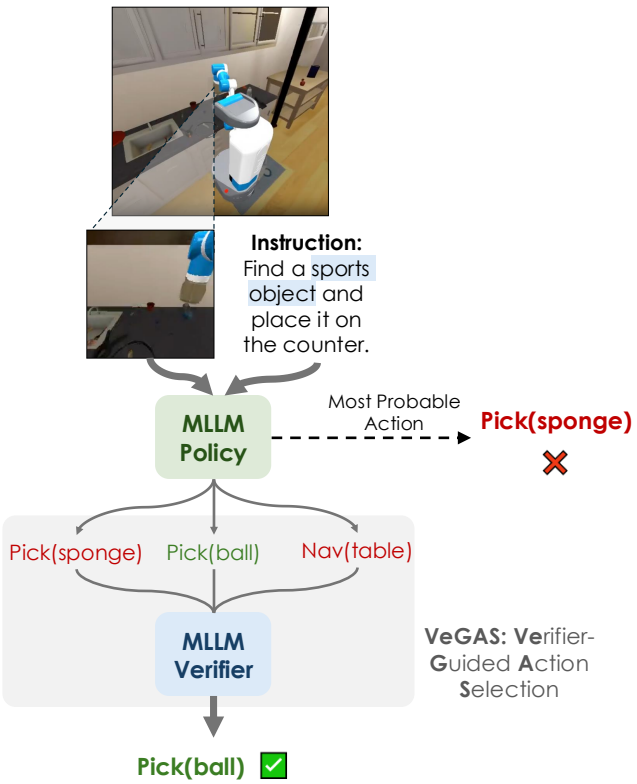

Fig. 1: **Overview of Verifier-Guided Action Selection (VEGAS).** Given a task instruction, standard policies decode a single action that may be incorrect under distribution shifts (right). VEGAS, samples multiple candidate actions with reasoning traces, evaluates them using a generative verifier, and executes the highest-scored action (bottom). This test-time verification substantially improves robustness in challenging out-of-distribution scenarios involving long-horizon tasks.

semantic task progression from egocentric observations alone, with compounding errors over longer horizons. Yet verification for such embodied reasoning is largely unexplored.

To bridge this gap, we introduce **Verifier-Guided Action Selection (VEGAS)**, a framework that improves the robustness of MLLM-based embodied agents by incorporating an explicit verification step at test time. Concretely, at each timestep VEGAS samples multiple candidate actions from the policy, each accompanied by a Chain-of-Thought rationale. A learned generative verifier [20], [21] then evaluates each candidate by producing an explicit reasoning trace followed by a correctness judgement, and the agent executes only the highest-scoring action (see Figure 1 for an overview). A critical finding is that using an MLLM as a verifier off-the-shelf yields no improvement over the base policy — general-purpose language understanding alone is insufficient for embodied verification. This motivates specialized verifier training; however, standard embodied datasets contain only successful demonstrations, providing no signal for what constitutes an incorrect action. To address this, we introduce an LLM-driven pipeline that automatically synthesizes diverse, realistic failure trajectories paired with verification annotations, constructing a rich curriculum of both correct and incorrect examples without additional human data collection.

VEGAS yields consistent improvements on embodied reasoning benchmarks [22], [8], [23], raising average success rates from 65% to 71% on LangR and from 44% to 49% on EB-ALFRED over strong CoT baselines, while significantly improving larger off-the-shelf policies.
Our key contributions are:

1) We propose Verifier-Guided Action Selection (VEGAS), a test-time verification framework for high-level embodied reasoning that samples diverse candidate actions and uses a learned generative verifier to select the most reliable one at each timestep. We find that using MLLMs off-the-shelf as verifiers does not improve performance, motivating our specialized training pipeline.
2) Training a verifier requires demonstrations of both correct and incorrect actions, yet embodied datasets typically lack the latter. To address this, we introduce an automated, LLM-driven pipeline that synthesizes diverse and realistic failure trajectories paired with verification annotations, without additional human data collection.
3) Extensive experiments in Habitat 2.0 [22], [8] and AI2-THOR [23] show that VEGAS consistently improves over strong CoT baselines, scales more effectively with test-time compute than Self-Consistency [24], and generalizes to large, off-the-shelf policies.

II. PRELIMINARIES

Problem Formulation. We formulate the agent's task as a sequential decision-making problem under partial observability. The agent's objective is to generate a sequence of actions $a_1, \ldots, a_T$ to accomplish a goal specified as a natural language instruction I (e.g., "Bring an item that can be used for cutting to the left counter"). At each timestep t, the agent receives an egocentric RGB image o_t as its observation. The agent's true underlying state s_t is not directly accessible. The agent must decide on its next action a_t based on the goal I and its history h_t composed of all its past observations and actions $(o_1, a_1, ..., o_{t-1}, a_{t-1}, o_t)$. Our aim is to learn a policy π that maps the goal and history to the next action: $\pi(a_t|I, o_{1:t}, a_{1:t-1})$. The action space A consists of high-level semantic actions, such as `pick(apple)` and `navigate(table)`. Following prior work [8], [6], we assume an oracle low-level policy that executes these high-level actions once selected.

Policy Architecture. We instantiate the policy π as a multimodal large language model (MLLM) that takes visual and text tokens as input and autoregressively generates text tokens as output. Given the goal in the form of a text instruction I and the history h_t as input, the policy autoregressively emits an output token sequence $y_t = (c_t, a_t)$. Here, c_t is an optional chain-of-thought rationale (a possibly empty sequence of text tokens) followed by the action token sequence a_t. Following [12], the actions are encoded in natural language (e.g., "`pick(apple)`"), and the output can be extracted to obtain the action a_t, which is sent to the environment to be executed by the low-level policy [8], [6].

Policy Training. We train the policy via imitation learning on expert demonstrations $\mathcal{D} = \{\tau\}$, where each trajectory $\tau = \big(I, (o_1, a_1), \ldots, (o_T, a_T)\big)$ depicts a successful execution of the task. The model is fine-tuned via supervised next-token prediction to maximize the likelihood of the expert output $y_t = (c_t, a_t)$, computing the loss only over output tokens, including the CoT prefix c_t when present.

III. VEGAS: VERIFIER-GUIDED ACTION SELECTION

The core idea of VEGAS is to augment a base policy with a learned verifier that, at each timestep, evaluates candidate actions and identifies the most reliable one before execution. Because off-the-shelf MLLMs fail as verifiers (as we will show in Sec. IV-A), we train a dedicated generative verifier on automatically synthesized failure data (Sec. III-A). Concretely, VEGAS operates via a Best-of-N procedure: at each timestep, the policy samples N candidate actions, the generative verifier [20], [21] evaluates each one by producing a verification reasoning trace followed by a correctness judgement, and the highest-scoring action is executed. Unlike discriminative verifiers that directly output a score [17], [25], generative verifiers think step-by-step before assigning a score, which has been shown to yield stronger performance while also making the scores more interpretable [20].

A. Synthetic Reasoning and Verification Data

CoT Augmentation. We start with a dataset of successful ('+') trajectories, $\mathcal{D}^+ = \{\tau^+\}$, where a trajectory τ^+ consists of the instruction I, and interleaved observations o and actions a, $\tau^+ = \{I, o_1, a_1^+, o_2, a_2^+, ...\}$. A model trained on these trajectories will directly output the next action given the observation. However, research in language reasoning and embodied AI has shown that thinking step-by-step can significantly improve the reasoning abilities of

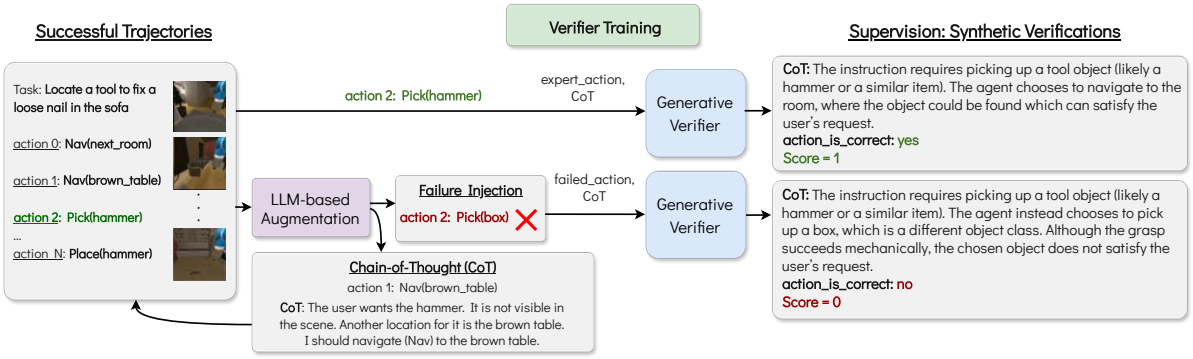

Fig. 2: **Synthetic data generation and training workflow for the verifier.** Successful trajectories are first processed by an LLM to produce chain-of-thought rationales for each action. Then, an LLM introduces realistic and diverse errors into these trajectories and annotates every action with a verification. This dataset is used to train a verifier through supervised finetuning.

models [14], [15]. To train embodied agents that can think step-by-step, similarly to [16], we prompt a teacher LLM (e.g., OpenAI o3) to augment every action with a chain-of-thought reasoning, c_i^+, explaining why the agent should perform the expected action a_i^+ given the previous inputs $I, o_1, a_1^+, ..., o_i$. This gives us a new dataset $\mathcal{D}_{CoT}^+ = \{\tau_{CoT}^+\}$, with $\tau_{CoT}^+ = \{I, o_1, (c_1^+, a_1^+), o_2, (c_2^+, a_2^+), ...\}$. Note that this procedure only augments every action a_i^+ with a chain-of-thought, and does not change the sequence of actions in the trajectories. Unlike [16], which grounds reasoning traces in visual features such as object and gripper positions for fine-grained manipulation, we target high-level semantic reasoning for tasks requiring long-horizon planning and linguistic interpretation, yielding $\mathcal{D}_{CoT}^+$.

Synthetic Failures for Verifier Training. To train a verifier, we require examples of both correct and incorrect actions. Because existing datasets rarely include failed executions, we introduce an automated and scalable pipeline that synthesizes unsuccessful trajectories. For each successful trajectory τ^+, we prompt a large language model (e.g., OpenAI o3) to produce a corresponding failed trajectory τ^-. The model generates realistic and diverse mistakes that span a broad range of failure modes in challenging scenarios, including *wrong object* (e.g., bringing an apple when the task requires a banana), *wrong receptacle* (e.g., placing an item on the sofa instead of the bed), and *precondition violation* (e.g., attempting to turn on a microwave without opening it first). We provide examples of synthetically generated incorrect actions in Figure 2.For both τ^+ and τ^-, we prompt the model to annotate every action with a verification consisting of chain-of-thought reasoning and a final binary judgement of the form `action_is_correct: yes/no`. These annotated samples provide the supervision used to train the verifier.

B. Verifier Training and Inference

We fine-tune an MLLM as a verifier that takes as inputs the instruction I, all previous actions $a_1, a_2, ..., a_{t-1}$, and the current observation o_t, chain-of-thought c_t, and action a_t sampled from the policy. It outputs a verification v_t consisting of a chain-of-thought and a verdict. The verifier is trained via supervised finetuning on the data described in Sec. III-A using the same next-token prediction objective as the policy (Sec. II). This process is illustrated in Figure 2.

During inference, at time t, we sample N candidate actions from the policy: $(c_t^{(1)}, a_t^{(1)}), (c_t^{(2)}, a_t^{(2)}), ..., (c_t^{(N)}, a_t^{(N)})$. We pass each candidate $(c_t^{(n)}, a_t^{(n)})$ to the verifier and, following the original GenRM procedure [20], sample M verifications per action to reduce variance, each consisting of a verification chain-of-thought and a verdict. The verdict can be mapped to a score ('`yes`' $\rightarrow$ 1 and '`no`' $\rightarrow$ 0), giving us M scores per action. We average these scores to obtain a final score for every action, $\sigma_t^{(n)}$. Finally, we select the highest-scoring action (Best-of-N): $a_t = \text{argmax}_{n \in [N]}[\sigma_t^{(n)}]$. The selected action a_t is then executed in the environment, and the process repeats at the next timestep.

IV. Experiments

A. Verifiers Improve Generalization Only When Finetuned

We analyze the effect of verification on the LangR benchmark (Table I). Fine-tuning with CoT supervision achieves 65% average success rate, surpassing prior state-of-the-art SemLang [12] and establishing a strong baseline.

We then evaluate whether verification can further improve the CoT policy. Using the same Qwen2.5-VL-3B-Instruct model as a zero-shot verifier (+ ZS Verifier) does not meaningfully improve over the CoT baseline and, in fact, slightly hurts average performance (64% vs. 65%). This shows that the Best-of-N selection paradigm alone is insufficient; without task-specific training, the verifier cannot reliably distinguish correct actions from incorrect ones. In contrast, equipping the CoT policy with our finetuned verifier (VEGAS) raises performance to 71%, with consistent gains across all task categories. The improvements are particularly pronounced in challenging scenarios such as *Multiple Objects*, where VEGAS provides roughly a 36% relative improvement over CoT alone and doubles performance compared to No-CoT. These results demonstrate that the gains of VEGAS stem not from sampling multiple candidates per se, but from our synthetic failure generation and verifier training pipeline.

To test whether these findings generalize beyond LangR, we repeat the evaluation on EB-ALFRED (Table II). Our CoT policy, obtained by fine-tuning Qwen2.5-VL-3B-Instruct, achieves an average success rate of 44%, as shown in Table II. This surpasses Qwen2.5-VL-72B-Instruct (success rate 30%),

Approach	Average	Paraphrastic Robustness				Behavioral Generalization			
		Rephrasing	Context	Irrelevant Text	Referring Expressions	Multiple Rearrange	Novel Objects	Multiple Objects	Conditional
Prior Work									
LLaRP (LLaMa-7B) [8]	46 ± 2	92 ± 2	34 ± 2	32 ± 2	26 ± 2	47 ± 5	95 ± 4	0 ± 1	39 ± 3
SemLang (LLaVA-1.5-7B) [12]	58 ± 1	92 ± 1	46 ± 14	66 ± 6	31 ± 3	80 ± 6	97 ± 0	2 ± 2	46 ± 4
Policy only (Qwen-2.5-VL-3B-Instruct)									
No-CoT	58	93	39	72	48	68	97	17	28
w/ CoT	65	98	50	85	59	64	97	25	42
w/ CoT policy + Verifier (Qwen-2.5-VL-3B-Instruct)									
+ ZS Verifier	64 ± 2	98 ± 1	50 ± 3	85 ± 5	48 ± 4	65 ± 4	97 ± 0	30 ± 4	40 ± 3
+ FT Verifier (VEGAS)	**71 ± 1**	**99 ± 1**	**52 ± 2**	**92 ± 1**	**62 ± 3**	**82 ± 1**	**97 ± 0**	**34 ± 2**	**48 ± 2**

TABLE I: **Success rates on the LangR [8] benchmark.** The No-CoT, w/ CoT, and VEGAS models are based on Qwen-2.5-VL-3B-Instruct. ZS and FT refer to zero-shot and finetuning, respectively. For verifier-based approaches, results are averaged over three runs. The No-CoT and CoT variants use greedy decoding for action selection, yielding deterministic outcomes. Our proposed approach combineing chain-of-thought reasoning with a finetuned verifier, consistently outperforms all baselines.

Approach	Average	Base	Common Sense	Complex Instructions	Long Horizon	Spatial	Visual Appearance
Finetuned Policies							
Qwen-3B w/ CoT	44	62	40	58	22	34	**48**
+Qwen-3B ZS Verifier	44 ± 2	64 ± 2	41 ± 7	53 ± 5	24 ± 3	35 ± 4	46 ± 2
+Qwen-3B FT Verifier (VEGAS)	**49 ± 2**	**67 ± 5**	**46 ± 2**	**62 ± 3**	**34 ± 2**	**43 ± 3**	41 ± 1
Gemma-4B w/ CoT	48	62	50	56	**28**	34	**56**
+Gemma-4B ZS Verifier	48 ± 2	66 ± 0	55 ± 1	61 ± 5	27 ± 3	32 ± 4	49 ± 6
+Gemma-4B FT Verifier (VEGAS)	**51 ± 1**	**67 ± 1**	**56 ± 0**	**67 ± 1**	25 ± 1	**37 ± 1**	53 ± 6
Zero-shot Policies							
Qwen-2.5-VL-72B	30	28	38	34	12	**38**	32
+Qwen-3B FT Verifier (VEGAS)	**38 ± 1**	**45 ± 3**	**39 ± 6**	**44 ± 7**	**15 ± 3**	36 ± 3	**47 ± 1**
Gemma-3-27B	19	24	24	26	0	**24**	18
+Qwen-3B FT Verifier (VEGAS)	**23 ± 0**	**30 ± 2**	**29 ± 1**	**27 ± 1**	**2 ± 2**	23 ± 1	**25 ± 1**
InternVL-3.5-38B	24	32	26	26	10	18	30
+Qwen-3B FT Verifier (VEGAS)	**35 ± 2**	**38 ± 2**	**36 ± 5**	**39 ± 5**	**27 ± 3**	**26 ± 2**	**41 ± 3**

TABLE II: **Success rates on EB-ALFRED [6].** ZS and FT refer to zero-shot and finetuning, respectively. The CoT policy employs greedy decoding, producing deterministic outcomes. Results involving verifiers are averaged over three runs. Our finetuned verifier consistently outperforms the CoT policy as well as the zero-shot verifier. Further, our finetuned verifier improves performance of larger policies, showing cross-model generalization.

a $\sim$20$\times$ larger model, and the best open-weights model reported on EB-ALFRED [6], thereby establishing a strong baseline. As on LangR, using the same model as a zero-shot verifier (+ ZS Verifier) does not improve over the CoT baseline (44% vs. 44%). In contrast, equipping the policy with our finetuned verifier (VEGAS) raises performance to 49%, confirming that task-specific verifier training is essential for effective verification. To assess whether these findings generalize beyond a single backbone, we repeat the experiment with Gemma-3-4B [26], fine-tuning both the CoT policy and the verifier. We observe the same trends: the zero-shot verifier provides no meaningful gains, while VEGAS improves the average success rate to 51%. This consistency across two different model families demonstrates that the benefits of VEGAS are not architecture-specific but arise from the finetuned verifier itself. Taken together, our results on LangR and EB-ALFRED demonstrate that verifying actions at test time can substantially enhance the generalization capabilities of embodied agents in challenging scenarios.

Verifier-Guided Improvement of Large Policies. We evaluate whether a small, finetuned verifier can improve large policies it was never trained with — a practical setting where large models are inaccessible for fine-tuning. We pair our Qwen2.5-VL-3B-Instruct verifier with several zero-shot policies on EB-ALFRED (Table II). Our verifier consistently improves every policy it is paired with; most notably, it improves Qwen-2.5-VL-72B—a model $\sim$20$\times$ its own size—from 30% to 38%, demonstrating that a compact verifier can meaningfully enhance policies far beyond its own scale.

V. Conclusion

We introduced **Verifier-Guided Action Selection (VEGAS)**, a test-time framework that improves the out-of-distribution robustness of embodied agents via an explicit verification step. Using an automated pipeline to synthesize failure trajectories for verifier training, VEGAS achieves consistent gains on LangR and EB-ALFRED, including over significantly larger off-the-shelf policies. Our analyses show that verifier finetuning is essential for reliably leveraging additional test-time compute.

ACKNOWLEDGEMENTS

Nishad Singhi is supported by a LOEWE Start-Professur (LOEWE/4b//519/05.01.002-(0006)/94). Marcus Rohrbach is supported in part by an Alexander von Humboldt Professorship in Multimodal Reliable AI, sponsored by Germany's Federal Ministry for Education and Research. For compute, we gratefully acknowledge support from the hessian.AI Service Center (funded by the Federal Ministry of Research, Technology and Space, BMFTR, grant no. 16IS22091) and the hessian.AI Innovation Lab (funded by the Hessian Ministry for Digital Strategy and Innovation, grant no. S-DIW04/0013/003). The work has benefited from the Excellence Cluster "Reasonable AI" by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-3057, DFG Emmy Noether Programme (CH 2676/1-1), European Union's Horizon Europe project "MANiBOT" (Grant No.: 101120823), European Union's Horizon Europe project "ARISE" (Grant No.: 101135959), BMFTR Project "RIG" (Grant No.: 16ME1001).

REFERENCES

[1] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[2] W. Li, Z. Yu, Q. She, Z. Yu, Y. Lan, C. Zhu, R. Hu, and K. Xu, "Llm-enhanced scene graph learning for household rearrangement," in *SIGGRAPH Asia 2024*, 2024.

[3] Y. Qiao, W. Lyu, H. Wang, Z. Wang, Z. Li, Y. Zhang, M. Tan, and Q. Wu, "Open-nav: Exploring zero-shot vision-and-language navigation in continuous environment with open-source llms," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[4] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[5] 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.

[6] R. Yang, H. Chen, J. Zhang, M. Zhao, C. Qian, K. Wang, Q. Wang, T. V. Koripella, M. Movahedi, M. Li *et al.*, "Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents," in *International Conference on Machine Learning (ICML)*, 2025.

[7] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International conference on machine learning*. PMLR, 2022, pp. 9118–9147.

[8] A. Szot, M. Schwarzer, H. Agrawal, B. Mazoure, R. Metcalf, W. Talbott, N. Mackraz, R. D. Hjelm, and A. T. Toshev, "Large language models as generalizable policies for embodied tasks," in *International Conference on Learning Representations (ICLR)*, 2023.

[9] G. Sarch, Y. Wu, M. Tarr, and K. Fragkiadaki, "Open-ended instructable embodied agents with memory-augmented large language models," in *ACL Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

[10] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[11] Y. Yang, T. Zhou, K. Li, D. Tao, L. Li, L. Shen, X. He, J. Jiang, and Y. Shi, "Embodied multi-modal agent trained by an llm from a parallel textworld," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 26 275–26 285.

[12] A. Szot, B. Mazoure, H. Agrawal, R. D. Hjelm, Z. Kira, and A. Toshev, "Grounding multimodal large language models in actions," *Advances in Neural Information Processing Systems*, vol. 37, pp. 20 198–20 224, 2024.

[13] S. Zhai, H. Bai, Z. Lin, J. Pan, P. Tong, Y. Zhou, A. Suhr, S. Xie, Y. LeCun, Y. Ma *et al.*, "Fine-tuning large vision-language models as decision-making agents via reinforcement learning," *Advances in neural information processing systems*, vol. 37, pp. 110 935–110 971, 2024.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[15] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo, "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[16] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine, "Robotic control via embodied chain-of-thought reasoning," in *8th Annual Conference on Robot Learning*, 2024.

[17] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[18] B. Brown, J. Juravsky, R. Ehrlich, R. Clark, Q. V. Le, C. Ré, and A. Mirhoseini, "Large language monkeys: Scaling inference compute with repeated sampling," *arXiv preprint arXiv:2407.21787*, 2024.

[19] C. Snell, J. Lee, K. Xu, and A. Kumar, "Scaling llm test-time compute optimally can be more effective than scaling model parameters," *arXiv preprint arXiv:2408.03314*, 2024.

[20] L. Zhang, A. Hosseini, H. Bansal, M. Kazemi, A. Kumar, and R. Agarwal, "Generative verifiers: Reward modeling as next-token prediction," in *International Conference on Learning Representations (ICLR)*, 2025.

[21] Z. Ankner, M. Paul, B. Cui, J. D. Chang, and P. Ammanabrolu, "Critique-out-loud reward models," *arXiv preprint arXiv:2408.11791*, 2024.

[22] A. Szot, A. Clegg, E. Undersander, E. Wijmans, Y. Zhao, J. Turner, N. Maestre, M. Mukadam, D. Chaplot, O. Maksymets, A. Gokaslan, V. Vondrus, S. Dharur, F. Meier, W. Galuba, A. Chang, Z. Kira, V. Koltun, J. Malik, M. Savva, and D. Batra, "Habitat 2.0: Training home assistants to rearrange their habitat," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[23] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749.

[24] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[25] F. Yu, A. Gao, and B. Wang, "Ovm, outcome-supervised value models for planning in mathematical reasoning," in *Findings of the Association for Computational Linguistics: NAACL 2024*, 2024, pp. 858–875.

[26] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, R. Merhej, S. Perrin, T. Matejovicova, A. Ramé, M. Rivière, L. Rouillard, T. Mesnard, G. Cideron, J. bastien Grill, S. Ramos, E. Yvinec, M. Casbon, E. Pot, I. Penchev, G. Liu, F. Visin, K. Kenealy, L. Beyer, X. Zhai, A. Tsitsulin, R. Busa-Fekete, A. Feng, N. Sachdeva, B. Coleman, Y. Gao, B. Mustafa, I. Barr, E. Parisotto, D. Tian, M. Eyal, C. Cherry, J.-T. Peter, D. Sinopalnikov, S. Bhupatiraju, R. Agarwal, M. Kazemi, D. Malkin, R. Kumar, D. Vilar, I. Brusilovsky, J. Luo, A. Steiner, A. Friesen, A. Sharma, A. Sharma, A. M. Gilady, A. Goedeckemeyer, A. Saade, A. Feng, A. Kolesnikov, A. Bendebury, A. Abdagic, A. Vadi, A. György, A. S. Pinto, A. Das, A. Bapna, A. Miech, A. Yang, A. Paterson, A. Shenoy, A. Chakrabarti, B. Piot, B. Wu, B. Shahriari, B. Petrini, C. Chen, C. L. Lan, C. A. Choquette-Choo, C. Carey, C. Brick, D. Deutsch, D. Eisenbud, D. Cattle, D. Cheng, D. Paparas, D. S. Sreepathihalli, D. Reid, D. Tran, D. Zelle, E. Noland, E. Huizenga, E. Kharitonov, F. Liu, G. Amirkhanyan, G. Cameron, H. Hashemi, H. Klimczak-Plucińska, H. Singh, H. Mehta, H. T. Lehri, H. Hazimeh, I. Ballantyne, I. Szpektor, I. Nardini, J. Pouget-Abadie, J. Chan, J. Stanton, J. Wieting, J. Lai, J. Orbay, J. Fernandez, J. Newlan, J. yeong Ji, J. Singh, K. Black, K. Yu, K. Hui, K. Vodrahalli, K. Greff, L. Qiu, M. Valentine, M. Coelho, M. Ritter, M. Hoffman, M. Watson,

M. Chaturvedi, M. Moynihan, M. Ma, N. Babar, N. Noy, N. Byrd, N. Roy, N. Momchev, N. Chauhan, N. Sachdeva, O. Bunyan, P. Botarda, P. Caron, P. K. Rubenstein, P. Culliton, P. Schmid, P. G. Sessa, P. Xu, P. Stanczyk, P. Tafti, R. Shivanna, R. Wu, R. Pan, R. Rokni, R. Willoughby, R. Vallu, R. Mullins, S. Jerome, S. Smoot, S. Girgin, S. Iqbal, S. Reddy, S. Sheth, S. Põder, S. Bhatnagar, S. R. Panyam, S. Eiger, S. Zhang, T. Liu, T. Yacovone, T. Liechty, U. Kalra, U. Evci, V. Misra, V. Roseberry, V. Feinberg, V. Kolesnikov, W. Han, W. Kwon, X. Chen, Y. Chow, Y. Zhu, Z. Wei, Z. Egyed, V. Cotruta, M. Giang, P. Kirk, A. Rao, K. Black, N. Babar, J. Lo, E. Moreira, L. G. Martins, O. Sanseviero, L. Gonzalez, Z. Gleicher, T. Warkentin, V. Mirrokni, E. Senter, E. Collins, J. Barral, Z. Ghahramani, R. Hadsell, Y. Matias, D. Sculley, S. Petrov, N. Fiedel, N. Shazeer, O. Vinyals, J. Dean, D. Hassabis, K. Kavukcuoglu, C. Farabet, E. Buchatskaya, J.-B. Alayrac, R. Anil, Dmitry, Lepikhin, S. Borgeaud, O. Bachem, A. Joulin, A. Andreev, C. Hardin, R. Dadashi, and L. Hussenot, “Gemma 3 technical report,” 2025. [Online]. Available: https://arxiv.org/abs/2503.19786