

Vision-Language Model Router for Robotics

Stanford CS224N Custom Project

Jadelynn Dao, Satvik Sharma, and Milan Ganai

Department of Computer Science

Stanford University

{jadelynn, satvik, mganai}@stanford.edu

Abstract

The scaling of Vision-Language Models (VLMs) has created a significant trade-off between reasoning capability and real-time responsiveness. While frontier models provide high-level planning for complex robotics tasks, their inference latency acts as a "stalling" bottleneck that disrupts human-robot interaction and poses physical hazards. This project investigates a Latency-Aware VLM Router that dynamically delegates embodied tasks to specific models on the continuum of lightweight "fast" models to a heavy "reasoning" models. Utilizing the VLABench dataset, we demonstrate that a substantial "reasoning gap" exists across skill domains like tool use and multi-step planning. Our preliminary findings show that an Oracle Router can achieve the success rate ceiling of frontier models while significantly reducing latency by offloading reactive tasks to smaller models.

1 Key Information to include

- Mentor: Caroline Choi
- External Collaborators (if you have any): No
- Sharing project: No
- Late Days (if you are using any):

2 Introduction

Vision-language models (VLMs) have rapidly expanded the capabilities of embodied agents, enabling robots to interpret natural-language instructions, reason about visual scenes, and generate multi-step action plans. These advances are especially promising for long-horizon manipulation tasks, where agents must integrate perception, commonsense knowledge, and sequential decision making over extended horizons. However, the same scaling trends that improve reasoning performance also introduce a practical systems bottleneck: larger, more capable models incur substantially higher inference latency. In robotics, this latency is not merely a matter of monetary cost or user experience. Slow planning can reduce responsiveness in closed-loop settings, increase safety risks when physical actions are delayed, weaken robustness to perturbations, and ultimately degrade performance in dynamic environments. As a result, deploying a single frontier-scale model for every embodied task may be unnecessarily expensive and operationally brittle, even if it achieves the highest average success rate.

Motivated by this tradeoff, our goal is to match the success rate of the most reasoning-intensive VLMs while minimizing the inference cost required to achieve that success. A key observation is that embodied tasks are not uniformly difficult: some demand long-horizon reasoning, tool use, or fine-grained spatial understanding, while others can be solved reliably by smaller and faster models. This heterogeneity creates an opportunity to reduce inference cost without sacrificing task performance through selective model allocation. While prior VLM routing work has exploited such heterogeneity in general multimodal settings, robotics has largely explored this trade-off through explicit hierarchical architectures, in which a stronger model is used for high-level semantic reasoning

and a smaller policy or controller is used for low-level execution. These hierarchies improve efficiency, but they typically rely on fixed role assignments or predetermined inference schedules, limiting their ability to fully exploit the Pareto frontier between latency and performance across diverse embodied tasks.

To this end, we propose a Latency-Aware VLM Router for robotics, a mixture-of-experts framework that selects among models with different speed-capability trade-offs. Given a high-level instruction and an initial egocentric image, the router uses a lightweight classifier over frozen multimodal embeddings to choose a model with minimal routing overhead. We also define an Oracle Router that selects the fastest successful model for each task, providing an upper bound on the gains achievable through routing. Unlike prior VLM routing work, we focus on embodied decision making, where latency directly affects action execution and where task difficulty is shaped by skill composition. By casting model selection itself as a routing problem for embodied reasoning, our approach moves beyond fixed hierarchical decompositions that are common in robotics [1, 2] and instead adapts the choice of VLM to the demands of each task instance. Our contributions are twofold: (1) we introduce a latency-aware routing framework for embodied long-horizon manipulation that dynamically selects among VLMs with different reasoning and latency profiles; and (2) we formulate an Oracle Router that identifies the fastest successful model for each task, enabling principled measurement of routing headroom and the Pareto frontier between task success and inference latency.

3 Related Work

Vision-Language Models for Robotics. Vision-language models (VLMs) have been used in robotics in three main ways. First, they can serve as generators of intermediate representations that make downstream control more tractable. Rather than predicting actions directly, these methods use VLMs to extract structured, robot-usable information from visual observations and language, such as affordances, keypoints, programs, or constraints. CLIPort combines open-vocabulary semantic grounding with spatially precise manipulation [3]; Instruct2Act synthesizes executable code for perception and action [4]; VoxPoser produces grounded 3D value maps from language-conditioned affordances and constraints [5]; RoboPoint predicts language-conditioned image-space keypoints [6]; and ReKep formulates relational keypoint constraints that can be optimized into robot motions [7]. In these systems, the VLM primarily functions as a semantic interface that translates high-level multimodal understanding into explicit representations that can be consumed by planners and controllers. Second, VLMs can be used more directly as high-level planners for long-horizon tasks. In this setting, the model reasons over task structure, scene context, and user intent across multiple decision steps, producing subgoals or action sequences for downstream execution. PaLM-E demonstrates that embodied multimodal models can jointly process visual observations, language, and robot state for sequential manipulation reasoning [8], while Hi Robot uses a hierarchical VLM-based framework to interpret open-ended instructions and human feedback, select the next subtask, and delegate execution to lower-level controllers [1]. Third, VLMs have increasingly been adopted as semantic backbones for vision-language-action models, where pretrained multimodal representations are adapted for action generation. RT-2 shows that co-fine-tuning vision-language models on robot trajectories can transfer web-scale semantic knowledge into robotic control [9], and OpenVLA further demonstrates the effectiveness of pretrained multimodal backbones for open, generalist manipulation policies with strong language grounding [10]. Our work further extends the use of VLMs in robotics by moving beyond the use of a single VLM to maximize long-horizon task performance, and instead studying Pareto-optimal trade-offs between inference latency and performance.

Routers for Vision-Language Models. Recent work on routing for vision-language models (VLMs) studies how to dispatch each query to the most suitable model in a heterogeneous pool in order to improve the trade-off between performance and latency relative to any single fixed model. In particular, VL-RouterBench formalizes this setting for VLMs and evaluates routers under joint quality, cost, and throughput constraints, showing that routing can recover better operating points along the Pareto frontier while still remaining below oracle selection [11]. Complementary multimodal routing methods further develop practical mechanisms for this trade-off, dynamically allocating queries across strong and weak or cloud and edge VLMs to balance accuracy, latency, and resource usage [12, 13]. In this setting, the goal is to identify Pareto-optimal operating points that maximize task performance while minimizing inference latency and other computational costs. In robotics, related ideas have mostly appeared in the form of hierarchical VLM architectures rather than explicit

routing policies: systems such as YAY Robot, Hi Robot, and HAMSTER separate high-level semantic reasoning from low-level control to improve long-horizon execution efficiency [14, 1, 2]. However, these approaches rely on fixed architectural decompositions or predetermined inference schedules, which limits their ability to adaptively optimize the latency–performance trade-off across embodied tasks. Our work extends routing for VLMs to embodied robotics by explicitly studying how to route long-horizon decisions so as to approach the Pareto frontier between task success and inference latency.

4 Approach

We study *latency-aware routing* for embodied vision-language model (VLM) inference. The goal is to decide, from the instruction and initial visual observation alone, when a fast instruction-following model is sufficient and when a slower reasoning-oriented model is worth the extra inference cost. Let (x_i, v_i) denote an embodied task instance consisting of a language instruction x_i and an initial egocentric RGB observation v_i . Let $\mathcal{M}$ be a models with different performances and latencies. For each example i and model m , we observe a task score $S_{i,m} \in [0, 100]$ and a latency $L_{i,m} \in \mathbb{R}_+$. A routing policy $\pi(x_i, v_i)$ selects one model for each example with the objective of improving the score–latency trade-off relative to any fixed standalone policy.

4.1 Latency-Aware VLM Router

Our router uses a frozen multimodal representation of the instruction-image pair together with lightweight linguistic difficulty features. Concretely, the input feature vector contains (1) a 512-dimensional CLIP-based fused text-image embedding, and (2) a small metadata vector including instruction length, token entropy, and average negative log-likelihood from a lightweight proxy language model. This representation is designed to capture both *semantic content* and *difficulty cues*. Intuitively, tasks involving clutter, distractors, long-horizon structure, or semantic ambiguity may benefit from assigning the input to a more capable, but potentially slower, model.

Rather than predicting the absolute score of each candidate model independently, we train the router to estimate the *relative utility* of different models in the candidate set $\mathcal{M}$. Let

$$u_{i,m} = U(x_i, v_i, m) \quad (1)$$

denote the utility of assigning example i to model $m \in \mathcal{M}$, where the utility is defined to capture the desired score–latency trade-off. The router learns a scoring function

$$f_\theta(x_i, v_i, m) \approx u_{i,m}, \quad (2)$$

and selects the model with the highest predicted utility:

$$\pi(x_i, v_i) = \arg \max_{m \in \mathcal{M}} f_\theta(x_i, v_i, m). \quad (3)$$

This formulation naturally induces a family of operating points on the score–latency frontier by varying the utility definition or decision rule to favor higher performance or lower latency. In this way, the router can adaptively allocate each task to the model in $\mathcal{M}$ that is predicted to offer the best trade-off for that particular input.

4.2 Oracle Router

To estimate the headroom available to routing, we define an oracle that chooses the best model for each task instance using hindsight. In the two-model setting, the oracle selects the higher-scoring model and breaks ties in favor of the faster one:

$$m_i^* = \arg \max_{m \in \mathcal{M}} (S_{i,m}, -L_{i,m}). \quad (4)$$

Although this oracle is not deployable, it provides an upper bound on the best score–latency trade-off achievable through perfect per-instance model selection.

4.3 Baselines

We compare against three baselines: a *standalone frontier model*, which runs the strongest high-capacity model on every task and serves as a performance upper bound among fixed policies; a

standalone lightweight model, which runs the fastest model on every task and serves as a latency-oriented lower bound; and a *random router*, which assigns each task to a model uniformly at random and tests whether the learned router captures meaningful task-dependent structure. In our primary comparison on VLABench, we instantiate these baselines using the Qwen3-VL 30B family.

4.4 Implementation Details

To reduce overfitting on a relatively small embodied benchmark, we use a compact supervised pipeline. We first apply PCA to the combined feature representation and retain the top 100 principal components. We then train a Random Forest model with 200 trees and maximum depth 12 to approximate the routing utility function over the candidate model set $\mathcal{M}$. Given an input (x_i, v_i) , the router evaluates the predicted utility of each candidate model and selects the one with the highest score. In our experiments, we instantiate this framework with two candidate models. All routing metrics are reported using 5-fold cross-validation so that each evaluated example is routed by a model that was trained without access to that example.

5 Experiments

5.1 Data

We evaluate on VLABench v1.0, a benchmark of 675 embodied task instances spanning six domains: *CommonSense* ($N=120$), *Complex* ($N=75$), *M&T* ($N=135$), *PhysicsLaw* ($N=90$), *Semantic* ($N=135$), and *Spatial* ($N=120$). These domains differ substantially in reasoning burden, ranging from relatively direct grounding tasks to settings requiring stronger semantic disambiguation, long-horizon planning, or physical reasoning.

This domain diversity is important for routing. If all tasks had similar reasoning requirements, then a single fixed model in $\mathcal{M}$ would be sufficient. Instead, the benchmark contains a mix of easier and harder examples, creating an opportunity for adaptive model selection. The domain-level score and latency breakdown in Figure 1 supports this view: stronger models generally achieve higher scores, but the magnitude of that advantage varies across domains, and so does the value of paying additional latency.

5.2 Evaluation Method

Our evaluation focuses on the trade-off between task performance and inference latency. For each task instance i and each model $m \in \mathcal{M}$, we record a benchmark score $S_{i,m} \in [0, 100]$ and a wall-clock latency $L_{i,m}$ in seconds. We compare four classes of policies: a static policy that always selects the lowest-latency model in $\mathcal{M}$, a static policy that always selects the highest-performing model in $\mathcal{M}$, a random routing baseline that samples a model from $\mathcal{M}$ uniformly at random, and the oracle routing policy defined in Section 3, which chooses the best model for each task using hindsight.

For each policy π , we report the average score $\frac{1}{N} \sum_{i=1}^N S_{i,\pi(x_i,v_i)}$ and the average latency $\frac{1}{N} \sum_{i=1}^N L_{i,\pi(x_i,v_i)}$. Following Figure 2, we also summarize routing quality relative to the random-routing baseline using two derived metrics: *score gain*, which measures vertical improvement in score at matched latency, and *latency decrease*, which measures horizontal reduction in latency at matched score. These metrics capture the two main ways routing can help: a routing policy may achieve a higher score under a similar latency budget, or it may achieve a similar score more quickly.

To better understand where routing opportunity comes from, we additionally analyze per-domain win/tie/loss behavior using the ± 5 point threshold shown in Figure 2. In our two-model experimental setting, this analysis compares the stronger and weaker standalone models on each example and identifies where their performance gap is large enough to matter and where they are effectively interchangeable.

5.3 Experimental Details

Although the routing framework in Section 4 is defined for a general candidate set $\mathcal{M}$, in our experiments we instantiate it with two models, namely $\mathcal{M} = \{m_1, m_2\}$, where m_1 is Qwen3-30B-

Table 1: Policy Performance Comparison

Policy	Avg Score	Avg Latency (s)	Score Gain	Latency Decrease
Best Policy	39.65	23.29	+1.79 pts	-1.27s
Efficiency Focus	35.98	21.19	+1.06 pts	-0.75s

Instruct and m_2 is Qwen3-30B-Thinking. Empirically, m_1 serves as the lower-latency model and m_2 serves as the higher-performing but slower model.

The router uses the features described in Section 4 to predict a utility value for each model in $\mathcal{M}$ and selects the model with the highest predicted utility. We reduce the feature representation to 100 dimensions with PCA and train a Random Forest with 200 trees and maximum depth 12. All results are reported with 5-fold cross-validation.

From the resulting routing frontier, we highlight two operating points in Table 1 and Figure 2. The first, denoted **Best Policy**, emphasizes score improvement. The second, denoted **Efficiency Focus**, favors lower latency while maintaining positive gain over the random-routing baseline.

5.4 Results

Figure 1 shows that, in our two-model instantiation of $\mathcal{M}$, Qwen3-30B-Thinking achieves higher average score than Qwen3-30B-Instruct across all six domains. The largest score gaps appear in *Complex*, *Spatial*, and *Semantic*, while *CommonSense*, *M&T*, and *PhysicsLaw* also show a clear but somewhat smaller advantage. This indicates that the benefit of assigning an input to a stronger model is broad rather than confined to a single task type. However, the same figure also shows that this improvement comes with a substantial latency cost. Qwen3-30B-Instruct typically operates in the single-digit to low-teens second range, while Qwen3-30B-Thinking ranges from roughly the high-20s to mid-50s seconds depending on domain. The largest latency burden appears in *Complex*, where the slower model is most expensive. Thus, always selecting the highest-performing model in $\mathcal{M}$ is effective in score terms but expensive in latency terms.

The oracle bars in Figure 1 show that this trade-off is not fixed. Across all domains, oracle latency is visibly lower than the latency of always selecting Qwen3-30B-Thinking, while oracle score remains close to, and in some cases slightly above, its score. This implies that many tasks do require stronger reasoning, but many others do not. In other words, there is genuine headroom for routing within $\mathcal{M}$.

Figure 2 provides a finer-grained explanation. Using a ± 5 point threshold, Qwen3-30B-Thinking wins on the majority of examples in every domain, with the strongest dominance in *Complex*. At the same time, several domains, especially *CommonSense*, *M&T*, and *PhysicsLaw*, have a non-trivial tie region. Those ties are exactly the cases where a router can save latency by selecting the cheaper model without sacrificing much score. The lower-latency model also retains a small but non-zero win rate across domains, which further shows that fixed use of the stronger model is inefficient on some instances.

Table 1 and Figure 2 summarize the learned routing results. The **Best Policy** achieves an average score of 39.65 at 23.29 seconds, corresponding to a score gain of +1.79 points and a latency decrease of $-1.27s$ relative to the random-routing baseline. The **Efficiency Focus** policy achieves an average score of 35.98 at 21.19 seconds, with a score gain of +1.06 and a latency decrease of $-0.75s$. Both policies improve over random routing, showing that the router extracts useful signals about which model in $\mathcal{M}$ is worth using for a given input.

Finally, Figure 3 illustrates a representative qualitative example. In a cluttered scene with multiple similar objects, Qwen3-30B-Instruct selects the wrong object and fails the task, while Qwen3-30B-Thinking correctly grounds the instruction and completes the action sequence, yielding a +25.0 point score gap. This example helps explain why routing matters in practice: the value of selecting a stronger model is especially large when the task requires semantic disambiguation, distractor rejection, and multi-step planning rather than purely reactive grounding.

Qwen3 30b Comparison: Performance and Latency Analysis

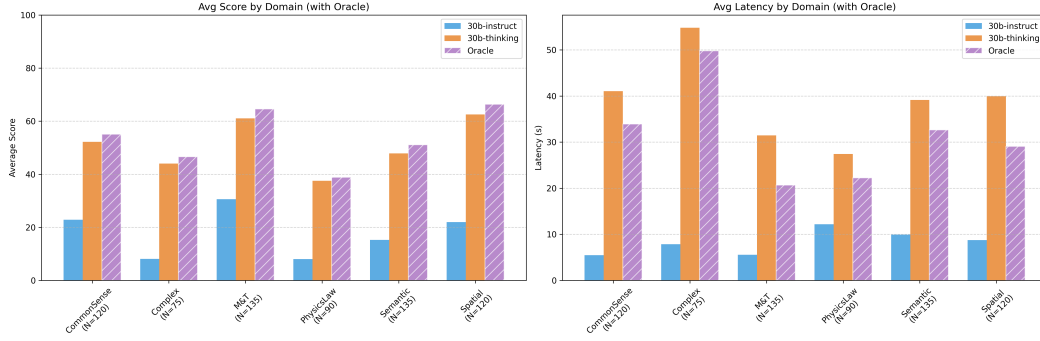


Figure 1: Qwen3 30b Comparison: Performance and Latency Analysis across six VLABench domains. Left: average score by domain for Qwen3-30B-Instruct, Qwen3-30B-Thinking, and the Oracle Router. Right: average latency by domain for the same policies.

Qwen3 30b Comparison: Win Rate Analysis

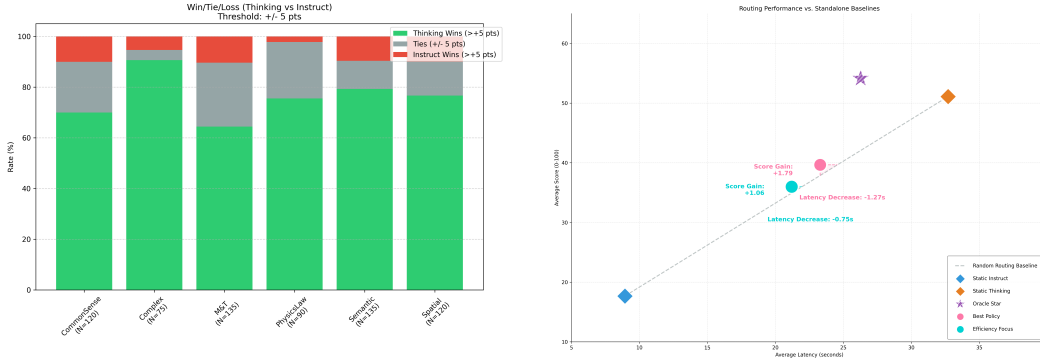


Figure 2: **Left:** Qwen3 30b Comparison: Win Rate Analysis. Stacked win/tie/loss breakdown for Thinking vs. Instruct under a ± 5 point threshold. **Right:** Routing Performance vs. Standalone Baselines. The learned routing policies are compared against static Instruct, static Thinking, random routing, and the Oracle Router in the score–latency plane.

6 Analysis

The results in Figure 1 establish a clear reasoning gap between the two standalone VLMs. Qwen3-30B-Thinking is consistently stronger than Qwen3-30B-Instruct across all domains, which suggests that embodied tasks benefit substantially from extra deliberation. However, the gap is not uniform. It is especially pronounced in *Complex* and *Spatial*, while domains such as *CommonSense* and *PhysicsLaw* show more room for ties and therefore more opportunities for a fast expert to substitute in without catastrophic loss. This non-uniformity is what makes routing meaningful. If the Thinking model dominated by a large margin on nearly every example, routing would collapse to the trivial policy of always choosing the stronger model. Conversely, if both models were usually tied, routing would reduce to pure latency minimization. Our results lie between these extremes: the stronger model is broadly better, but not uniformly necessary.

The oracle in Figure 1 shows that the main inefficiency in static Thinking is not poor performance, but overuse of expensive reasoning. The oracle remains close to Thinking on score while achieving noticeably lower latency across all six domains. That pattern implies that many examples do need a stronger model, but many others do not.

The domains that appear most promising for routing are those where Thinking has a real average advantage but the win/tie/loss plot in Figure 2 still shows substantial tie mass. In particular, *M&T* and



Figure 3: **Qualitative example** showing a cluttered multimodal manipulation task. Qwen3-30B-Instruct fails by selecting the wrong object, while Qwen3-30B-Thinking completes the correct multi-step plan, yielding a +25.0 point score gap.

PhysicsLaw look like domains where an accurate router could often save latency without giving up much score. By contrast, *Complex* appears much less forgiving: Thinking dominates so consistently there that routing should usually escalate to the stronger model. Figure 2 shows that the learned router recovers part of the oracle opportunity. This is the expected outcome for a difficult prediction problem: the router must decide, from only the initial image and instruction, whether extra reasoning will translate into a meaningful downstream performance gain. The Best Policy prioritizes intelligence gain, moving upward relative to the random baseline at the cost of somewhat higher latency than the efficiency-focused operating point. The Efficiency Focus policy, by contrast, stays closer to the low-latency region while still outperforming random routing. Together, these two points demonstrate that routing does not produce a single fixed answer; instead, it exposes a tunable frontier that lets a practitioner decide how much latency they are willing to spend for additional task success.

Figure 3 illustrates a qualitative, representative failure case for the fast model. The instruction is "Put the apple on the plate," but the scene contains multiple visually similar distractors in a cluttered workspace. Qwen3-30B-Instruct incorrectly reaches for the orange and fails to complete the task, obtaining a score of 65.0 at 21.2 seconds. Qwen3-30B-Thinking instead identifies the correct target, executes the full action sequence successfully, and achieves a score of 90.0 at 38.5 seconds. The figure highlights this as a +25.0 point intelligence gap. This example clarifies what routing must detect. The difficulty is not simply raw perception, since both models observe the same input. Rather, the stronger model appears better at combining language grounding, distractor rejection, and sequential planning under ambiguity. These are precisely the cases where spending additional latency on reasoning is worthwhile.

Taken together, Figures 1, 2, and 3 support a consistent conclusion: reasoning-heavy VLMs are substantially better for embodied tasks, but they are also substantially slower; the gap is large enough to matter operationally, and non-uniform enough to make routing worthwhile. The oracle confirms that there is real headroom for adaptive model selection, and the learned router recovers part of that headroom by outperforming random mixtures on both score and latency.

7 Conclusion

This work is a first step toward understanding whether routing is useful for embodied vision-language model inference. Our results suggest that it is: stronger reasoning-oriented models achieve better task performance, but at a substantial latency cost, and this trade-off varies enough across tasks to make

adaptive model selection worthwhile. The oracle reveals clear routing headroom, and the learned router recovers part of that headroom by improving over random mixtures on both score and latency. Our study also has important limitations. We consider a small candidate set with only two models from the same family, use a one-shot routing decision from the initial image and instruction, and evaluate only in an offline benchmark setting. As a result, we do not yet capture broader deployment considerations such as closed-loop execution, safety, or user interaction quality. Our routing model is also fairly simple and would be interesting to explore larger routers and if the inference cost of the router becomes a consideration. Future work should study routing over larger and more diverse model pools, allow routing decisions to adapt over time as new observations arrive, and evaluate these methods in more realistic embodied settings. Overall, our findings suggest that routing is a promising direction for allocating compute adaptively in embodied systems.

Team contributions (Required for multi-person team)

All members adequately contributed to the final project.

Jadelynn Dao: Originally thought of the idea, led experimental design, and created figures.

Satvik Sharma: Part of ideation discussions, led formulation and analysis.

Milan Ganai: Part of ideation discussions, worked on experimentation, and helped with writing.

References

- [1] Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, Adrian Li-Bell, Danny Driess, Lachy Groom, Sergey Levine, and Chelsea Finn. Hi robot: Open-ended instruction following with hierarchical vision-language-action models, 2025.
- [2] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Raymond Yu, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation, 2025.
- [3] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation, 2021.
- [4] Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model, 2023.
- [5] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023.
- [6] Wentao Yuan, Jiafei Duan, Valts Blukis, Wilbert Pumacay, Ranjay Krishna, Adithyavairavan Murali, Arsalan Mousavian, and Dieter Fox. Robopoint: A vision-language model for spatial affordance prediction for robotics, 2024.
- [7] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation, 2024.
- [8] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023.
- [9] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspier Singh, Anikait Singh, Radu Soricut,

- Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.
- [10] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model, 2024.
- [11] Zhehao Huang, Baijiong Lin, Jingyuan Zhang, Jingying Wang, Yuhang Liu, Ning Lu, Tao Li, and Xiaolin Huang. Vl-routerbench: A benchmark for vision-language model routing, 2025.
- [12] Mingyuan Wu, Jize Jiang, Haozhen Zheng, Meitang Li, Zhaocheng Li, Beitong Tian, Bo Chen, Yongjoo Park, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Cache-of-thought: Master-apprentice framework for cost-effective vision language model reasoning, 2025.
- [13] Xin Tang, Youfang Han, Fangfei Gou, Wei Zhao, Xin Meng, Yang Yu, Jinguo Zhang, Yuanchun Shi, Yuntao Wang, and Tengxiang Zhang. Ecvl-router: Scenario-aware routing for vision-language models, 2025.
- [14] Lucy Xiaoyang Shi, Zheyuan Hu, Tony Z. Zhao, Archit Sharma, Karl Pertsch, Jianlan Luo, Sergey Levine, and Chelsea Finn. Yell at your robot: Improving on-the-fly from language corrections, 2024.

A Appendix (optional)

All major inclusions are in the main paper.