FINAL REPORT — FRA503

GRPO-Based Reinforcement Learning for Flow-Matching VLA Models

Applying Group Relative Policy Optimization to SmolVLA for Improved Generalization in Robotic Manipulation

Jessada Pranee **65090500405**

Chinavat Nachaithong **65090500408**

2025

Abstract

We investigate the application of Group Relative Policy Optimization (GRPO) to SmolVLA [7], a compact Vision-Language-Action model employing a flow-matching action head, with the objective of improving task success rates on the LIBERO-10 benchmark [4] beyond the supervised fine-tuning (SFT) baseline. Through six iterative experiments totalling approximately 6,000 training steps per run, we find that GRPO fine-tuning fails to improve upon the SFT baseline (51.50% mean success rate) under the constraints of this investigation. We identify three compounding failure modes: simulation throughput bottleneck, binary reward signal collapse (all-fail / all-success degeneracy), and insufficient training horizon. Critically, we observe a strong inverse correlation between policy weight drift and task success rate across experiments — policies that deviate more from the pretrained checkpoint consistently perform worse — suggesting that the reward signal, as implemented, is insufficient to guide the policy toward genuinely better behavior. We report these negative results alongside systematic ablations of KL regularization, reward shaping, and replay-based interventions, and argue that at least 100,000 training steps with vectorized simulation are likely necessary for meaningful convergence.

1. Introduction

Recent progress in Vision-Language-Action (VLA) models has demonstrated strong capabilities in robotic manipulation, primarily driven by supervised imitation learning on large-scale demonstration datasets. Models such as RT-2 [9], OpenVLA [2], π_0 [1], and SmolVLA [7] achieve compelling in-distribution performance, but the resulting policies are brittle — they fail to generalize to novel object configurations or unseen environments.

SimpleVLA-RL [3] has shown that reinforcement learning fine-tuning with GRPO can improve VLA generalization beyond pure imitation. However, these experiments are conducted on autoregressive discrete token action heads, leaving flow-matching architectures unexplored. This distinction matters: flow-matching heads generate actions through a continuous denoising process rather than discrete token sampling, making the application of GRPO non-trivial.

This report presents the results of applying GRPO to SmolVLA's flow-matching head, evaluated on the LIBERO-10 benchmark. Our investigation was structured as an iterative debugging process across six experiments, each motivated by the failure modes of its predecessor. While we do not demonstrate improvement over the SFT baseline, we contribute a systematic characterization of the failure modes specific to this architecture and training regime, and identify the conditions under which success may be achievable with greater computational resources.

Summary of Contribution

We present the first systematic attempt at GRPO fine-tuning of a flow-matching VLA model. Our primary contribution is a diagnostic analysis of why this approach fails under constrained training budgets, rather than a performance improvement, providing a foundation for future work in this direction.

2. Background and Related Work

2.1 The Generalization Problem in Imitation-Learned VLA Models

Imitation learning from expert demonstrations has been the dominant training paradigm for robotic manipulation. While this produces strong in-distribution performance, the resulting policies are brittle — they lack the ability to adapt to novel object configurations or transfer to unseen environments. SimpleVLA-RL [3] formalizes this observation and proposes RL as a corrective mechanism. However, its experiments are conducted on autoregressive token-based action heads, leaving flow-matching architectures unexplored.

2.2 Group Relative Policy Optimization (GRPO)

GRPO is a policy gradient variant originally proposed in DeepSeekMath [6] for large language model reasoning. Unlike PPO, which requires a separately trained value function, GRPO estimates the advantage of each sample relative to the group mean from a single batch of rollouts. For a group of n outputs sampled from the same context, the advantage of output i is:

$$\hat{A}_i = \frac{r_i - \text{mean}(r_1, \ldots, r_n)}{\text{std}(r_1, \ldots, r_n) + \epsilon} \tag{1}$$

This formulation eliminates the need to train a critic network, which is notoriously difficult to stabilize over high-dimensional continuous observation spaces. A key property of Eq. (1) is that when all trajectories in a group receive the same reward (all-success or all-fail), the advantage collapses to zero and no gradient update is applied — a degenerate case we encounter frequently in our experiments.

2.3 SmolVLA and the Flow-Matching Action Head

SmolVLA [7] is a compact VLA model built on the SmolVLM [5] vision-language backbone. Unlike VLA models that decode actions as discrete language tokens, SmolVLA employs a flow-matching head that models the action distribution as a continuous probability path. At inference time, actions are generated by integrating a learned vector field from a sampled Gaussian noise vector $z \sim \mathcal{N}(0, I)$ over N Euler steps:

$$x_{t+\Delta t} = x_t + \Delta t \cdot v_\theta(x_t, t), \qquad \Delta t = \frac{1}{N} \tag{2}$$

A critical property is that for any fixed N and initial noise z, this mapping is fully deterministic. Stochasticity in the output is controlled entirely by the choice of z, which motivates our independent noise sampling strategy for generating trajectory groups.

3. Proposed Method

3.1 Training Pipeline Overview

The proposed training pipeline follows the standard GRPO loop: sample a group of n trajectories from the same observation context, evaluate each trajectory against the task reward, compute group-relative advantages via Eq. (1), and update the policy via a clipped

policy gradient objective. The key adaptation for flow-matching is in how trajectory diversity is induced.

3.2 Trajectory Sampling via Independent Noise

Because SmolVLA's flow-matching head is deterministic given a fixed noise vector, diversity across the n group trajectories is induced by sampling n distinct Gaussian noise vectors at the start of each rollout:

$$\tau_i = \texttt{Execute}(\texttt{FlowDenoise}(\pi_\theta(o, l),\ z_i)), \qquad z_i \sim \mathcal{N}(0, I) \tag{3}$$

Each noise vector z_i produces a distinct action sequence, which is then executed in the simulation environment to produce a full trajectory and a binary task success reward $r_i \in \{0, 1\}$.

3.3 Two-Tier Denoising Rollout Design

The final pipeline design (introduced in Experiment 2) employs two denoising tiers within each group. The first slot is always a 1-step denoised trajectory (producing structurally noisy, failure-prone actions), while all subsequent slots use 10-step fully denoised trajectories. The 1-step reward is weighted by 0.1 to discount the contribution of actions that may succeed by chance due to noise rather than learned policy:

$$R(\tau_i) = \begin{cases} 0.1 \cdot r(\tau_i) & \text{if } i = 1 \text{ (1-step denoising)} \\ 1.0 \cdot r(\tau_i) & \text{if } i > 1 \text{ (10-step denoising)} \end{cases} \tag{4}$$

3.4 GRPO Objective with KL Regularization

The policy is updated via the clipped GRPO objective with a KL-divergence penalty relative to the SFT checkpoint.

$$\mathcal{L}_{\text{GRPO}} = -\,\mathbb{E}\Big[\min\Big(\rho_i(\theta)\,\hat{A}_i,\ \text{clip}(\rho_i(\theta),\,1-\epsilon,\,1+\epsilon)\,\hat{A}_i\Big)\Big] + \beta\, D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}) \tag{5}$$

where $\rho_i(\theta) = \pi_\theta(\tau_i)/\pi_{\theta_{\text{old}}}(\tau_i)$ is the importance sampling ratio, ϵ is the clipping threshold, β is the KL penalty coefficient, and $D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$ represents the KL divergence between the current policy and the reference SFT checkpoint. Fine-tuning is applied jointly to the flow-matching head (higher learning rate) and the vision-language backbone (lower learning rate), with the SFT checkpoint retained as the KL reference.

4. Experimental Setup

4.1 Environment and Benchmark

All experiments are conducted in the LIBERO simulation environment [4], using the LIBERO-10 task suite. Each task requires a robot arm to complete a multi-step manipulation objective specified by a natural language instruction. The simulation is implemented in MuJoCo [8]. Each policy is evaluated over 20 rollouts per task.

4.2 Baseline

The SFT baseline is the pre-trained SmolVLA(500M parameters) model fine-tuned on LIBERO-10 demonstration data via behavioral cloning. This produces a mean success rate of 51.50% across the 10 tasks, serving as the reference for all subsequent experiments.

4.3 Compute Constraints

A significant structural constraint of this work is simulation throughput. Each full trajectory (10-step denoising, 20 action chunks) requires 10–20 seconds of wall-clock simulation time on CPU, as MuJoCo's step function is not GPU-accelerated despite GPU rendering support. A single GRPO training step consequently requires 30–60 seconds. This limits the total training budget to approximately 6,000 steps per experiment, corresponding to roughly 17–33 hours of wall-clock time. Scaling to the 100,000 steps likely needed for convergence would require approximately 280–550 hours per run, highlighting vectorized simulation as a prerequisite for future work.

5. Experiments

The six experiments below are presented as an iterative diagnostic sequence. Each experiment is motivated by a specific failure mode identified in the previous one.

5.1 Experiment 1: Initial Three-Horizon Pipeline

The initial pipeline sampled three trajectories per group using denoising step counts of 8, 9, and 10. The composite reward was a weighted average across all three horizons (weights 0.1, 0.2, 0.7 respectively). Three failure modes were identified:

- **Throughput:** Three full simulations per group tripled the per-step cost to 30–60 seconds, making large-scale training infeasible.
- **Reward degeneracy:** Binary rewards produced frequent all-fail or all-success groups, collapsing the advantage signal to zero.
- **Learning rate instability:** The initial learning rates (head: 10^{-3}, backbone: 10^{-4}) caused excessively large weight updates. After the first update, the policy failed on all tasks and never recovered.

5.2 Experiment 2: Two-Tier Rollout with Reduced Learning Rate

The pipeline was redesigned to use two denoising tiers per group: one 1-step trajectory and five 10-step trajectories, eliminating the 8- and 9-step horizons. This reduced per-step cost to 10–20 seconds. Learning rates were also reduced substantially. The 1-step trajectory was intended to reliably produce failed cases, providing a structural source of reward diversity within each group.

This design partially mitigated the all-success degeneracy, but the all-fail case remained when the policy encountered difficult tasks.

5.3 Experiment 3: KL Divergence Analysis

Initial observations during training revealed that an excessively high learning rate led to highly negative loss values, which drove the policy's weights too far from their initial state

and subsequently prevented the model from discovering further success cases. To mitigate this issue, the learning rate was substantially reduced. This adjustment successfully enabled the model to maintain success cases across training steps, although the "all-fail" degeneracy still intermittently occurred during challenging steps.

To investigate the underlying training dynamics, the KL-divergence was monitored over a full 6,000-step run. The analysis revealed that the total weight deviation of the fine-tuned policy was only 0.17% relative to the pretrained SFT checkpoint, indicating a structural shift of less than 1%. Consequently, the leading hypothesis is that the KL-divergence penalty exerts an overly restrictive regularizing effect, coercing the policy to preserve its original weights and inhibiting meaningful parameter updates required for adaptation.

5.4 Experiment 4: KL Divergence Ablation

The KL penalty was removed to test the hypothesis from Experiment 3. The resulting per-task success rates were identical to Experiment 2 (49.00% mean), with indistinguishable per-task breakdown. This demonstrates that KL regularization was not the bottleneck — the policy remained effectively frozen regardless of its presence.

Key Finding: KL Is Not the Bottleneck

Experiments 3 and 4 produced identical results, showing the policy fails to update meaningfully whether or not KL regularization is applied. The root cause is insufficient gradient signal, not over-regularization.

5.5 Experiment 5: Fallback Reward for All-Fail Groups

To address the all-fail degeneracy, a fallback mechanism was introduced: when an entire group fails, an additional 100-step denoised trajectory is executed. If this trajectory succeeds, it receives reward = 1; if it fails, it receives reward = 0.05 (a small nonzero value to provide at least minimal gradient signal).

The intervention backfired completely due to systemic *reward hacking*. Empirical analysis of the training logs revealed that approximately 67% of the training steps ended up triggering the fallback mechanism and failing (thus harvesting the 0.05 reward), while only 33% were driven by natural successes. Because this 0.05 penalty value yielded a non-negligible relative advantage compared to the true all-zero baseline of difficult tasks, the policy optimized for the easiest path to maximize rewards—joneing to preferentially produce failure-mode actions. As a result, the mean success rate dropped sharply to 30.00% and weight drift increased to 0.38%, confirming that the policy was actively moving away from the pretrained checkpoint toward a degenerate, failure-incentivizing mode.

5.6 Experiment 6: Success Replay Pool

A success replay pool was introduced to prevent all-fail degeneracy: previously observed successful trajectories were stored, and if a group produced only failures, a cached success trajectory was injected to break the degeneracy.

The pool grew during the first 80–300 training steps as the policy initially succeeded on easier tasks. However, as the policy drifted from the SFT checkpoint under continued training, the distribution of training data became heavily distorted. A granular breakdown of the training steps showed that 55% of the batch composition was driven by historical replay injections, 14% of the steps were skipped due to empty pool states on unaccomplished

tasks, and only 31% originated from actual, live successes. This heavy reliance on historical data led to severe *overfitting* to a narrow set of identical, previously cached trajectories. As the model progressively lost its generalization capabilities and the ability to generate new successes, the final evaluation showed a mean success rate of only 12.00% with a weight drift of 0.60% — marking the largest structural deviation and lowest success rate across all experiments.

6. Results

6.1 Per-Task Success Rates

Table 1 reports per-task success rates across the SFT baseline and all six experiments.

Table 1: Per-task success rates (%) on LIBERO-10 across all experiments. Task descriptions are truncated for space. Bold indicates the highest score per task. Tasks are indexed 0–9.

Task	Base	E1	E2	E3	E4	E5	E6
T0: alphabet soup & tomato sauce	**20**	**30**	**30**	20	**30**	20	0
T1: cream cheese & butter	**60**	40	55	**60**	55	40	0
T2: stove & moka pot	**85**	80	**85**	70	**85**	50	0
T3: black bowl in drawer	**95**	75	**95**	75	**95**	30	10
T4: white mug & yellow mug	25	20	15	**35**	15	10	25
T5: book to back compartment	70	50	70	**85**	70	45	35
T6: white mug & chocolate pudding	**30**	**30**	20	15	20	20	0
T7: alphabet soup & cream cheese	**40**	**40**	20	20	20	35	0
T8: both moka pots on stove	35	35	35	**45**	35	10	10
T9: yellow mug in microwave	55	**70**	65	65	65	40	40
Mean	**51.5**	47.0	49.0	45.0	49.0	30.0	12.0

6.2 Weight Drift and Success Rate Correlation

Table 2 reports the relationship between policy weight drift and final mean success rate for the experiments in which drift was measured. Weight drift is defined as the mean absolute percentage difference in parameter values between the fine-tuned model and the SFT checkpoint at the end of training. Drift measurements were not collected for Experiments 1 and 2, as instrumentation was added only after the KL degeneracy issue was identified in Experiment 3. Notably, Experiments 3 and 4 share an identical drift value of 0.17% despite one using KL regularization and the other not, indicating that the KL penalty had no measurable effect on how much the policy moved from its initialization.

Table 2: Policy weight drift relative to SFT checkpoint vs. mean success rate. A clear inverse correlation is observed: greater drift consistently corresponds to lower success.

Experiment	Weight Drift (%)	Mean Success (%)	Notes
SFT Baseline	0.00	51.50	Reference checkpoint
Experiment 3 (KL)	0.17	45.00	With KL regularization
Experiment 4 (no KL)	0.17	49.00	KL removed; identical result
Experiment 5	0.38	30.00	Fallback reward introduced
Experiment 6	0.60	12.00	Replay pool; catastrophic degradation

7. Analysis and Discussion

7.1 The Central Failure Mode: Reward Signal Insufficient to Guide Drift

The most important finding of this investigation is the inverse correlation between weight drift and success rate shown in Table 2. Under the SFT baseline, the policy encodes manipulation knowledge learned from thousands of expert demonstrations. The GRPO reward signal, as implemented here, is too noisy and sparse to identify a direction of weight change that improves task success. Any deviation from the SFT checkpoint therefore erodes this pretrained knowledge without replacing it with better behavior — the policy is being pushed off a local optimum with no better optimum within reach.

This creates a dilemma: the policy cannot improve without changing its weights, but any weight change under the current reward signal degrades performance. The resolution requires either a denser, more informative reward signal or sufficient training steps for the policy to explore beyond the degradation valley.

Core Observation

Under a binary task-success reward with group sizes of 6 and a training budget of ∼6,000 steps, the GRPO gradient signal is insufficient to guide the flow-matching policy toward improved behavior. Every experiment that induced greater weight change produced lower success rates.

7.2 Structural Mismatch Between GRPO and Robotic Simulation

GRPO was designed for large language model training, where sampling thousands of outputs per group is computationally cheap. In robotic simulation, each rollout requires

physical simulation at 500ms per step, resulting in 10–20 seconds per trajectory. This imposes three compounding constraints that are absent in the LLM setting:

1. **Small group size.** With 6 trajectories per group, the group-relative advantage estimate has high variance. In LLM applications, groups of 8–64 are common.
2. **Low task diversity per batch.** Each training step samples a single task. Rare tasks that could provide informative gradient signal are encountered infrequently.
3. **Insufficient total steps.** With 30–60 seconds per step, 6,000 steps represents the practical limit. SimpleVLA-RL [3] and comparable RL-for-robotics works train for orders of magnitude more environment interactions.

7.3 Reward Degeneracy: All-Fail and All-Success Collapse

Binary rewards interact poorly with GRPO's group-relative normalization. When a task is sufficiently difficult that the policy fails on all n trajectories in a group, the advantage is zero and no update occurs. Similarly, when a task is easy enough that all trajectories succeed, the advantage is again zero. In both cases, the training step is wasted.

Our training logs confirm this degeneracy in practice. Consecutive steps (e.g., Steps 62–67 in the provided logs) show all-fail patterns of the form $[0, 0, 0, 0, 0, 0.1]$, where only the 1-step trajectory occasionally succeeds by chance. This means the model receives no useful gradient for entire sequences of steps when encountering difficult tasks, which constitute the majority of the LIBERO-10 suite.

Attempts to mitigate this via a fallback reward (Experiment 5) and replay pool (Experiment 6) both worsened performance by introducing corrupted reward signals and catastrophic forgetting respectively.

7.4 KL Regularization Is Not the Bottleneck

The identical results between Experiments 3 and 4 (both 49.00% mean, identical per-task breakdown) provide a clean ablation demonstrating that KL regularization had no effect on the training outcome. The policy failed to move in either condition.

This is further supported by the training logs, which show KL values consistently at or below 10^{-4} and clipping ratios of 0.00 throughout training. The importance sampling ratio $\rho_i(\theta)$ never approached the clipping boundary, confirming that policy updates were vanishingly small regardless of regularization. The KL penalty was not constraining the policy — the gradient signal itself was insufficient to produce meaningful updates.

7.5 Task-Level Vulnerability

Examining per-task results in Table 1 reveals a consistent pattern: higher-difficulty tasks (T0, T4, T6) degrade earliest and most severely, while initially high-performing tasks (T3 at 95%, T2 at 85%) survive Experiments 1–4 but collapse catastrophically in Experiments 5–6. This structural collapse directly aligns with the data distribution anomalies observed in the later runs. In Experiment 5, the 67% prevalence of the fallback failure reward incentivized even high-performing tasks to drift toward degenerate reward-hacking behaviors. In Experiment 6, the 55% dominance of static replay injections forced the policy to overfit to a small subset of early, easy successes, completely wiping out its performance on both consistently hard and originally stable tasks. This suggests that the sparse reward signal

provides gradient information primarily for tasks near the decision boundary of success, offering little protection to the rest of the suite — which is precisely where balanced optimization is most critical.

8. Future Work

Based on the failure modes identified in this investigation, we propose the following directions for future work:

Vectorized simulation. The single most impactful improvement would be parallelizing environment rollouts using Isaac Lab [**?**] or similar GPU-accelerated simulators, reducing per-step cost by orders of magnitude and enabling the 100,000+ steps likely required for convergence.

Extended training horizon. Even within the current simulator, we estimate that 100,000 steps are necessary to observe meaningful policy improvement. The 6,000-step budget in this work is likely insufficient for any RL signal to overcome the initialization advantage of the SFT checkpoint.

Denser reward shaping. Binary task success provides a single bit of feedback per trajectory. Intermediate rewards — for example, rewarding object contact, correct gripper orientation, or proximity to goal state — would provide more informative gradient signal and reduce the all-fail degeneracy that wastes training steps on difficult tasks.

Larger group sizes. Increasing the group size from 6 to 16 or more would reduce advantage estimate variance. This requires either faster simulation or acceptance of longer per-step wall-clock times.

9. Conclusion

We presented a systematic investigation of GRPO-based reinforcement learning fine-tuning applied to SmolVLA's flow-matching action head on the LIBERO-10 benchmark. Across six experiments totalling approximately 6,000 training steps each, no configuration improved upon the SFT baseline (51.50% mean success rate). The primary finding is that the GRPO reward signal, under the constraints of binary rewards, small group sizes, and limited training steps, is insufficient to guide the flow-matching policy toward genuinely better behavior. Any weight change away from the SFT checkpoint degrades performance, as evidenced by a strong inverse correlation between weight drift and task success rate.

The three compounding failure modes — simulation throughput, reward signal collapse, and insufficient training horizon — are not fundamental to the approach, but reflect the practical constraints of this investigation. We believe that with vectorized simulation, extended training, and denser reward shaping, GRPO fine-tuning of flow-matching VLA models remains a promising direction worth pursuing.

References

[1] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. Physical Intelligence.

[2] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. 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. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713, 2025.

[3] Haozhan Li et al. Simplevla-rl: Scaling vla training via reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2026. Poster.

[4] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.

[5] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

[6] 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. *arXiv preprint arXiv:2402.03300*, 2024.

[7] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[8] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[9] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and

Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2165–2183, 2023.