SCEVLA: SPARSE COLLABORATIVE EXITS WITH SPECIALIZED TOKENS FOR DYNAMIC VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models have become a cornerstone of embodied intelligence, but their large-scale architectures introduce substantial inference latency, limiting real-time robotic deployment. Existing dynamic VLA approaches alleviate this issue by allocating computation adaptively. However, dense exit supervision and conflicting updates to shared parameters can impair intermediate predictions. In this work, we propose SceVLA: a dynamic-depth VLA framework that enables efficient inference via sparse collaborative exits with specialized tokens. SceVLA introduces sparsely distributed exits at different backbone depths, where each exit is equipped with dedicated action tokens to specialize in predictions at its corresponding representation level. To encourage effective cross-depth collaboration, we design a collaborative attention mechanism that allows exits to exploit complementary information from different layers, while blocking gradients through cross-exit feature transfers. Furthermore, we develop a lightweight router that dynamically selects an exit for each input through unidirectional attention over exit-specific tokens and backbone representations. On CALVIN, SceVLA achieves a $2.28\times$ policy-query speedup with AvgLen decreasing from 4.42 to 4.36. On RoboTwin 2.0, it achieves a $2.13\times$ speedup with 72.7% success, compared with 85.5% for the full-depth base policy, revealing a benchmark-dependent accuracy–latency trade-off.

1 INTRODUCTION

Vision-Language-Action (VLA) models, including RT-1 (Brohan et al., 2022), RT-2 (Zitkovich et al., 2023), OpenVLA (Kim et al., 2024), and π_0 (Black et al., 2025), integrate visual observations and language instructions to predict actions for robot control. Their increasing reliance on large backbones, however, incurs substantial inference latency, limiting real-time deployment. Recent work, such as DeeR-VLA (Yue et al., 2024), has shown that many situations encountered during a rollout require much less computation than the full model provides. The computational demands of action prediction vary with the current observation and task stage, motivating policies that adjust their inference depth at each step.

Dynamic inference offers a way to make this allocation adaptive. DeeR-VLA (Yue et al., 2024) equips a multimodal backbone with intermediate exits and uses action consistency to determine when computation can terminate. MoLe-VLA (Zhang et al., 2026) instead learns to activate selected backbone layers through a spatial-temporal router. These approaches establish that different robot states need not incur the same computational cost.

For multi-exit policies, jointly training intermediate predictors introduces two challenges. First, maintaining too many exits spreads limited training resources across numerous prediction objectives, even though such a dense set of exits may be unnecessary. As shown in Figure 1, the sparse three-exit configuration achieves higher AvgLen than denser configurations at their shared depths. Fewer exits need not narrow the minimum-to-maximum computation range: retaining a sufficiently wide span between the shallowest and deepest submodels preserves this range while reducing the number of intermediate choices. This suggests that training a small set of strong exits can be more effective than densely covering backbone depths.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Second, submodels that share a backbone can produce conflicting gradients during joint optimization. A shallow exit encourages intermediate features to support immediate action decoding, whereas a deeper exit may benefit from retaining information for subsequent processing. These competing demands can limit the quality of the shared representation and degrade the performance of individual exits. Together, these observations motivate a small set of well-trained exits with specialized action representations and controlled information exchange.

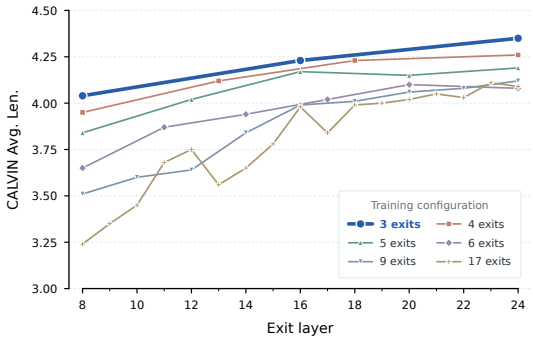

Figure 1: **Exit density on CALVIN.** Preliminary models are trained with different numbers of exits. Avg. Len. measures consecutively completed tasks (higher is better). Three exits at layers 8, 16, and 24 outperform denser configurations at shared depths while preserving the 8–24 depth span.

We introduce **SceVLA**, a dynamic-depth VLA framework built around *sparse collaborative exits with specialized tokens*. Our central idea is to give each exit its own action-prediction tokens while enabling structured information exchange across depths. SceVLA concentrates training on a sparse set of exits spanning shallow to deep backbone layers, with dedicated action tokens for each exit. These tokens provide an explicit interface for learning action predictions from the representation level available at that exit. At inference time, SceVLA can predict an action and stop at an early exit, or continue to a deeper exit when the current input requires more computation.

To connect these specialized interfaces, we develop a collaborative attention mechanism that enables exits to use complementary information across representation levels. The mechanism is designed to combine cues relevant to geometric interaction and semantic interpretation while limiting interference between exit-specific objectives. We further introduce a lightweight router that uses unidirectional attention over exit-specific tokens and backbone representations to select an exit for each input. Together, these components address three aspects of dynamic inference: producing actions at different depths, coordinating the representations used for those predictions, and choosing how much computation to allocate to the current decision.

We evaluate SceVLA on CALVIN (Mees et al., 2022) and RoboTwin 2.0 (Chen et al., 2025), covering language-conditioned long-horizon manipulation and bimanual manipulation, respectively. VLM-only SceVLA achieves $2.28\times$ and $2.13\times$ policy-query speedups, respectively, with a small AvgLen reduction on CALVIN and a larger success-rate reduction on RoboTwin. These results support the use of specialized, collaborative exits for balancing action quality and inference cost. Our main contributions are:

- We propose a sparse multi-exit VLA architecture that concentrates training on a small set of specialized action-token streams while retaining a broad computation range.
- We develop collaborative attention for cross-depth information exchange and a lightweight attention-based router for input-adaptive exit selection.
- We demonstrate over $2\times$ policy-query speedups on CALVIN and RoboTwin 2.0 and analyze the resulting accuracy–latency trade-offs.

2 Related Work

2.1 Vision-Language-Action Models

VLA models map visual and language representations to action prediction, providing a route from large-scale pretraining to language-conditioned robot control. RT-2 (Zitkovich et al., 2023) expresses robot actions as tokens and jointly trains on robotic trajectories and vision-language tasks, while OpenVLA (Kim et al., 2024) provides an open-source policy by fine-tuning a pretrained vision-language backbone on robot demonstrations. Beyond autoregressive action prediction, π_0 (Black et al., 2025) combines a pretrained vision-language model with flow matching to generate continuous actions. Action representation is another important design choice: FAST (Pertsch

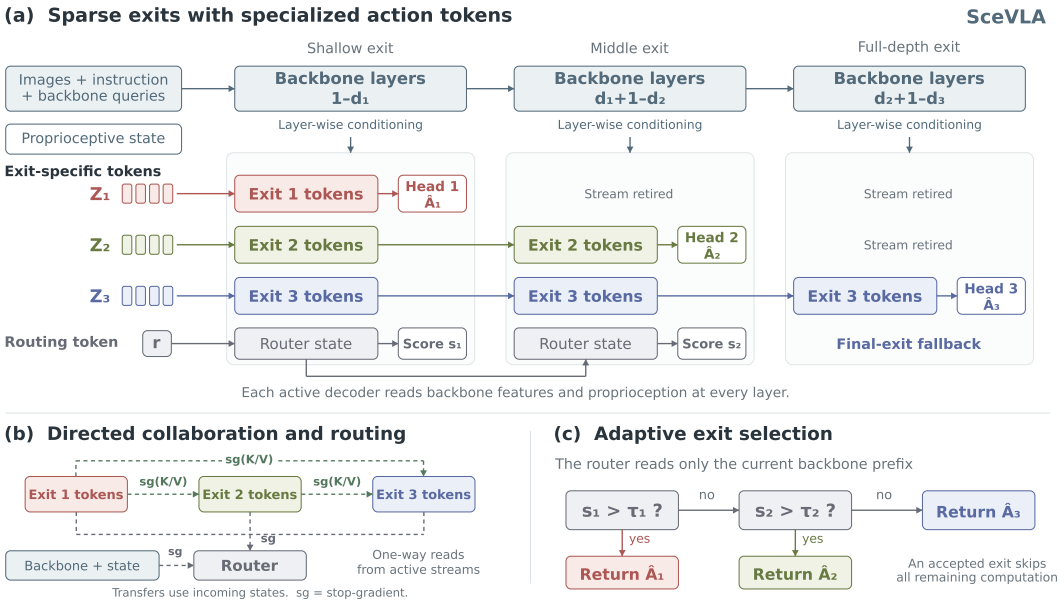

Figure 2: **Overview of SceVLA.** (a) Specialized action-token streams accumulate layer-wise backbone features and proprioceptive information, with independent action predictions at depths $d_1 < d_2 < d_3 = L$. The three exits are at layers 8/16/24 on CALVIN and 12/24/36 on RoboTwin 2.0. Each stream stops updating after its designated exit. (b) While streams are active, deeper-exit streams attend to shallower-exit states, and a separate routing token reads the backbone-conditioning features and active action streams. Dashed arrows denote detached feature transfers (sg: stop-gradient); all reads use incoming states at the current layer. (c) At each non-final exit, the router compares its stopping score s_j with threshold τ_j. The first accepted exit returns its action chunk and skips all remaining computation; the full-depth exit serves as the fallback.

et al., 2025) uses a compression-based, frequency-domain tokenizer to represent action sequences, enabling autoregressive learning from high-frequency robot trajectories. These developments establish multiple ways to connect multimodal representations to robot actions. SceVLA focuses on how this connection is learned at different backbone depths, using dedicated action tokens at selected exits to support dynamic computation.

2.2 Efficient VLA Inference

VLA acceleration methods target different sources of computational cost, including sequential action decoding and repeated visual processing. OpenVLA-OFT (Kim et al., 2025) studies the joint effects of action representation, decoding, and fine-tuning objectives, and combines parallel decoding with action chunking to improve inference efficiency. CEED-VLA (Song et al., 2025) uses consistency training to accelerate parallel Jacobi decoding and permits termination before full iterative convergence. Its early-exit decision concerns decoding iterations, which is a different computational dimension from the backbone depth selected in SceVLA. VLA-Cache (Xu et al., 2025) exploits temporal continuity by reusing cached key-value representations for minimally changed visual tokens while recomputing task-relevant tokens. Its layer-adaptive reuse strategy further controls how much token computation is retained at different depths. These approaches reduce the cost of generating an action sequence or processing successive observations. SceVLA instead allocates backbone computation to the current input through a set of specialized action-prediction exits.

2.3 Adaptive Depth and Multi-Exit Learning

Early-exit networks make inference cost input-dependent by attaching predictors to intermediate representations. BranchyNet (Teerapittayanon et al., 2016) introduces side-branch classifiers that allow confident predictions to terminate early. MSDNet (Huang et al., 2017) develops multi-scale features and dense connections to support accurate intermediate predictions under anytime and bud-

geted inference. These methods show that useful early predictions depend on representation design as well as a stopping criterion.

For robotic policies, DeeR-VLA (Yue et al., 2024) uses a multi-exit backbone, auxiliary action supervision, and action-consistency-based termination to adapt computation to the current situation. Layer-skipping approaches retain a different set of computation paths: MoLe-VLA (Zhang et al., 2026) uses spatial-temporal routing and cognitive self-distillation to activate selected layers, while DySL-VLA (Yang et al., 2026b) distinguishes consistently executed informative layers from skippable incremental layers and trains the resulting policy with skip-aware distillation. LoopVLA (Shen et al., 2026) takes a recurrent approach, repeatedly applying a shared Transformer block and learning when further refinement is unnecessary. Its sufficiency head uses dedicated tokens that attend to action representations to determine whether additional refinement is needed.

SceVLA combines a sparse set of exits with dedicated action-token streams and directed cross-exit collaboration. This design emphasizes the specialization and interaction of intermediate action predictors alongside adaptive depth selection.

3 METHOD

SceVLA couples a shared vision-language backbone with a small set of specialized action-prediction exits. Each exit maintains its own action-token stream, which accumulates backbone information up to its designated depth. Collaborative attention transfers information from shallower-exit streams to deeper-exit streams, while a separate routing token determines whether computation should terminate. We first describe the action architecture and then introduce routing and the two-stage training procedure. Figure 2 provides an overview of the framework.

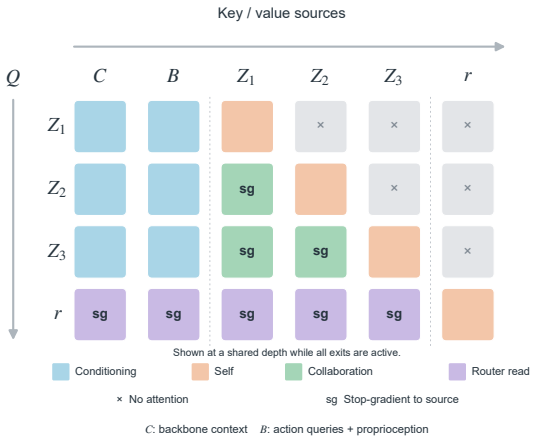

Figure 3: **Attention connectivity.** Rows are queries; columns are key/value sources. C: backbone context; B: action queries and proprioception; Z_j: exit tokens; r: router. Green denotes shallow-to-deep collaboration; purple denotes router reads. `sg` stops gradients to the source. Gray crosses mark absent connections. Shown with all exits active.

3.1 SPARSE EXITS WITH SPECIALIZED ACTION TOKENS

At time t, a policy π_θ maps an observation $\mathbf{o}_t = (\mathbf{I}_t, \ell_t, \mathbf{q}_t)$ to an action chunk $\mathbf{A}_t = [\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}] \in \mathbb{R}^{H \times d_a}$. Here, $\mathbf{I}_t$ denotes camera images, ℓ_t the language instruction, $\mathbf{q}_t$ the proprioceptive state, H the prediction horizon, and d_a the action dimension.

SceVLA places K sparsely distributed exits at ordered backbone depths $d_1 < \cdots < d_K = L$, where L is the full backbone depth. Keeping the shallowest and deepest exits far apart preserves a broad range of computation budgets, while avoiding a prediction objective at every intermediate layer. Each exit has its own learnable action tokens and decoder parameters. These tokens form a separate action-prediction stream that attends to backbone features and is supervised at its designated exit depth. We denote the token states for exit j at backbone depth m by $\mathbf{Z}_j^m$. At depth d_j, an exit-specific readout converts $\mathbf{Z}_j^{d_j}$ into the action chunk $\widehat{\mathbf{A}}_j$. This gives each exit a dedicated representation for decoding actions from the information available at its computation budget, while retaining a shared vision-language backbone.

3.2 COLLABORATIVE ATTENTION

Dedicated tokens enable specialization, but learning each exit in isolation would forgo useful information acquired by other exits. We therefore introduce directed collaboration: deeper-exit streams attend to the representations of shallower-exit streams while both are active. Figure 3 summarizes

the attention connections and their gradient flow. This lets a deeper stream incorporate features already useful for shallow action prediction and refine them with subsequent backbone features.

At backbone depth m, exit j uses its own tokens as attention queries. Its key-value memory combines its own token states, the backbone-conditioning features, and the token states of active shallower exits. Let $\mathcal{B}_j^m$ denote the key-value banks from the first two sources, and let $\mathcal{P}_j^m = \{i : i < j,\ m \le d_i\}$ index the available shallower exits. The collaborative memory is

$$\mathcal{M}_j^m = \mathcal{B}_j^m \cup \left\{(\mathrm{sg}(\mathbf{K}_i^m), \mathrm{sg}(\mathbf{V}_i^m)) : i \in \mathcal{P}_j^m\right\}, \tag{1}$$

where $\mathbf{K}_i^m$ and $\mathbf{V}_i^m$ are the donor stream's projected keys and values, and sg denotes stop-gradient. The memory banks are concatenated along the token dimension, and a single softmax jointly normalizes attention over all sources. The resulting features update the recipient's action tokens.

Stopping gradients at the transferred keys and values allows feature reuse without letting a deeper exit's loss change the donor representation through that connection. Each stream is thus trained for its own prediction depth while providing information to deeper streams; gradients through the shared backbone remain possible. Collaboration uses only states computed from the current backbone prefix. When a shallow stream reaches its exit depth, it stops updating, while deeper streams retain the information already incorporated into their states.

3.3 Unidirectional Routing and Adaptive Inference

A lightweight router decides whether the current exit is sufficient for the input. Its learnable routing token attends to the backbone-conditioning features and all currently active action-token streams. This combines input context with representations specialized for different prediction depths. Information flows only from the action streams to the router: action tokens never attend to the routing token, and the features supplied to the router are detached from the action policy. This asymmetry appears in the router row and column of Figure 3.

At each non-final exit j, a sigmoid readout of the routing token produces a stopping score $s_j \in (0, 1)$. If $s_j > \tau_j$, the policy returns $\widehat{\mathbf{A}}_j$ and skips the remaining backbone layers and action decoders. Otherwise, computation continues to the next exit; the deepest exit is the fallback. The router uses only features already computed at the current depth, including the current prefix states of streams assigned to deeper exits. It therefore makes stopping decisions without evaluating future backbone layers. The thresholds τ_j control the trade-off between action quality and inference cost.

3.4 Two-Stage Training

Action learning. We first train the action policy on demonstrations $\mathcal{D} = \{(\mathbf{o}, \mathbf{A})\}$, supervising every exit with the same normalized action chunk:

$$\mathcal{L}_{\mathrm{act}} = \mathbb{E}_{(\mathbf{o},\mathbf{A})\sim\mathcal{D}}\left[\frac{1}{K}\sum_{j=1}^{K}\frac{\|\widehat{\mathbf{A}}_j - \mathbf{A}\|_1}{Hd_a}\right]. \tag{2}$$

The loss assigns equal weight to all exits and averages absolute errors over action coordinates and timesteps. We optimize the specialized tokens and decoders, the proprioception projector, the backbone action queries, and the backbone's low-rank adaptation parameters. Direct supervision makes each exit usable for action prediction, while detached collaboration allows deeper streams to reuse shallower representations.

Router learning. We then freeze the action policy and train only the router. For each demonstration, let $e_j = \|\widehat{\mathbf{A}}_j - \mathbf{A}\|_F^2/(Hd_a)$ be exit j's mean squared action error. The target is the earliest exit whose error falls below its training tolerance ϵ_j, with the deepest exit as fallback:

$$j_{\mathrm{tar}} = \min\left(\{j < K : e_j < \epsilon_j\} \cup \{K\}\right). \tag{3}$$

With binary labels $y_j = \mathbf{1}[j_{\mathrm{tar}} = j]$, the router minimizes

$$\mathcal{L}_{\mathrm{route}} = \mathbb{E}_{\mathcal{D}}\left[\frac{1}{K-1}\sum_{j=1}^{K-1}\mathrm{BCE}(s_j, y_j)\right], \tag{4}$$

Table 1: **CALVIN ABC→D.** Chain-success rates (%) and AvgLen (0–5); higher is better. †RGB-D input. Reporting sources are detailed in Appendix A.

Method	SR_1	SR_2	SR_3	SR_4	SR_5	**AvgLen**
Published results						
RoboFlamingo (Li et al., 2024)	82.4	61.9	46.6	33.1	23.5	2.48
GR-1 (Wu et al., 2023)	85.4	71.2	59.6	49.7	40.1	3.06
OpenVLA (Kim et al., 2024)	91.3	77.8	62.0	52.1	43.5	3.27
RoboDual (Bu et al., 2025a)†	94.4	82.7	72.1	62.4	54.4	3.66
UniVLA (Bu et al., 2025b)	95.5	85.8	75.4	66.9	56.5	3.80
Seer (Tian et al., 2024)	94.4	87.2	79.9	72.2	64.3	3.98
MoDE (Reuss et al., 2024)	96.2	88.9	81.1	71.8	63.5	4.01
OpenVLA-OFT (Kim et al., 2025)	96.3	89.1	82.4	75.8	66.5	4.10
RoboVLMs (Li et al., 2026)	98.0	93.6	85.4	77.8	70.4	4.25
VLA-Adapter (Wang et al., 2025)	99.1	94.6	88.8	82.8	76.5	4.42
SceVLA: robot-pretrained						
Fixed layer 8	97.8	91.5	82.3	73.0	70.2	4.15
Fixed layer 16	96.4	93.4	84.6	76.7	73.2	4.24
Fixed layer 24	98.8	96.6	86.6	80.3	76.3	4.39
SceVLA: VLM-only						
Fixed layer 8	98.8	96.9	88.6	76.5	73.6	4.34
Fixed layer 16	98.4	95.4	87.1	75.3	73.1	4.29
Fixed layer 24	99.0	96.7	88.5	76.9	74.0	4.35

Here, BCE denotes binary cross-entropy, and each non-final exit is supervised on every training example. The error tolerances ϵ_j determine training labels; the stopping thresholds τ_j govern inference. This separation preserves the learned action predictions while fitting the router to their depth-dependent accuracy. Ground-truth actions are needed only during training. Benchmark-specific settings are summarized in Section 4.1, with training hyperparameters in Appendix A.

4 Experiments

We evaluate SceVLA along three axes: closed-loop manipulation performance, inference efficiency, and the contributions of specialized tokens, collaborative attention, and adaptive routing.

4.1 Experimental Setup

Benchmarks. We evaluate CALVIN (Mees et al., 2022) in the ABC→D setting on 1,000 instruction sequences, reporting chain-success rates and average completed sequence length (AvgLen). On RoboTwin 2.0 (Chen et al., 2025), we evaluate all 50 official tasks under the Clean setting with 100 rollouts per task and report mean task success.

Models and training. We build on VLA-Adapter (Wang et al., 2025) for CALVIN and Qwen-OFT (StarVLA Community, 2026) for RoboTwin, placing exits at layers 8/16/24 and 12/24/36, respectively. We compare *robot-pretrained* initialization from the corresponding action policies with *VLM-only* initialization without prior robot action training; both undergo the two-stage training procedure in Section 3. Controlled comparisons use matched training and evaluation settings, with latency measured on a single RTX 4090. Metric definitions, optimization settings, action-chunk configurations, and execution protocols are provided in Appendix A.

4.2 Manipulation Performance

Fixed-exit performance. On CALVIN (Table 1), robot-pretrained SceVLA improves from 4.15 to 4.39 AvgLen as the exit depth increases from 8 to 24, with the deepest exit approaching the 4.42 full-depth baseline. With VLM-only initialization, the shallow exit already reaches 4.34 AvgLen, compared with 4.35 at layer 24, indicating that substantial computation can be removed while retaining sequence-level performance. On RoboTwin 2.0 (Table 2), deeper exits consistently improve

Table 3: **Performance and inference efficiency.** Mean batch-one policy-query latency on an RTX 4090, with query frequency and speedup over the full-depth base policy. Score denotes CALVIN AvgLen or RoboTwin success (%). Frequency is $1000/t$ and speedup is t_{base}/t for latency t in ms. SceVLA (R/V) denotes adaptive inference with robot-pretrained/VLM-only initialization; paired fixed-exit scores are listed in this order, with identical timing and depth entries shared.

Policy	Score ↑	Latency (ms) ↓	Frequency (Hz) ↑	Speedup ↑	Avg. depth
CALVIN (VLA-Adapter)					
Base, full depth	4.42	157.1	6.37	1.00	24
DeeR-VLA (Yue et al., 2024)	4.14	111.5	8.97	1.41	12.8
MoLe-VLA (Zhang et al., 2026)	4.09	115.1	8.69	1.36	13.2
SceVLA, layer 8	4.15 / 4.34	64.3	15.55	2.44	8
SceVLA, layer 16	4.24 / 4.29	113.9	8.78	1.38	16
SceVLA, layer 24	4.39 / 4.35	157.1	6.37	1.00	24
SceVLA (R)	4.26	78.8	12.69	1.99	11.9
SceVLA (V)	4.36	68.9	14.51	2.28	8.1
RoboTwin 2.0 (Qwen-OFT)					
Base, full depth	85.5	169.4	5.90	1.00	36
DeeR-VLA (Yue et al., 2024)	63.3	101.0	9.90	1.68	17.1
MoLe-VLA (Zhang et al., 2026)	60.9	109.7	9.12	1.54	18.0
SceVLA, layer 12	61.5 / 65.3	67.5	14.81	2.51	12
SceVLA, layer 24	68.5 / 70.4	116.9	8.55	1.45	24
SceVLA, layer 36	82.6 / 77.6	169.4	5.90	1.00	36
SceVLA (R)	69.1	70.7	14.14	2.40	14.9
SceVLA (V)	72.7	79.6	12.56	2.13	16.3

success in both initialization settings: from 61.5% to 82.6% with robot-pretrained weights and from 65.3% to 77.6% with VLM-only weights. The stronger depth dependence on RoboTwin highlights the importance of retaining deeper computation for this benchmark.

Effect of initialization. VLM-only initialization yields stronger shallow exits on both benchmarks, improving CALVIN AvgLen by 0.19 and RoboTwin success by 3.8% over robot-pretrained initialization. Robot-pretrained initialization instead achieves the highest full-depth scores, particularly on RoboTwin (82.6% versus 77.6%). Initialization therefore affects shallow and full-depth performance differently.

Table 2: **RoboTwin 2.0 Clean.** Mean success (%) over 50 tasks. DP3 uses point clouds; the other methods use RGB.

Method	SR (%) ↑
ACT (Zhao et al., 2023)	29.7
Diffusion Policy (Chi et al., 2024)	28.0
DP3 (Ze et al., 2024)	55.2
RDT-1B (Liu et al., 2025)	34.5
π_0 (Black et al., 2025)	46.4
$\pi_{0.5}$ (Intelligence et al., 2025)	60.2
X-VLA (Zheng et al., 2025)	70.0
Qwen3VL-OFT (Yang et al., 2026a)	61.7
VLAct-OFT (Yang et al., 2026a)	80.5
SceVLA: robot-pretrained / VLM-only	
Fixed layer 12	61.5 / 65.3
Fixed layer 24	68.5 / 70.4
Fixed layer 36	82.6 / 77.6

4.3 Inference Efficiency and Routing Behavior

Adaptive performance and efficiency. Table 3 shows that adaptive SceVLA provides a better accuracy–latency trade-off than DeeR-VLA (Yue et al., 2024) and MoLe-VLA (Zhang et al., 2026) under both initialization settings. On CALVIN, VLM-only SceVLA reaches 4.36 AvgLen at 68.9 ms per query, yielding a $2.28\times$ speedup over the full-depth baseline with an AvgLen decrease of 0.06. It also slightly exceeds its own fixed layer-24 score (4.35), while using an average depth of 8.1. Robot-pretrained SceVLA reaches 4.26 AvgLen at 78.8 ms, improving on its fixed layer-16 score at lower latency. On RoboTwin, VLM-only SceVLA achieves 72.7% success at 79.6 ms, a $2.13\times$ speedup over the base policy; robot-pretrained SceVLA is faster at 70.7 ms but reaches 69.1% success. Both adaptive variants outperform their respective fixed middle exits while reducing latency. However, the RoboTwin speedups involve a larger performance trade-off: VLM-only adaptive inference remains 12.8% below the 85.5% full-depth base policy. These re-

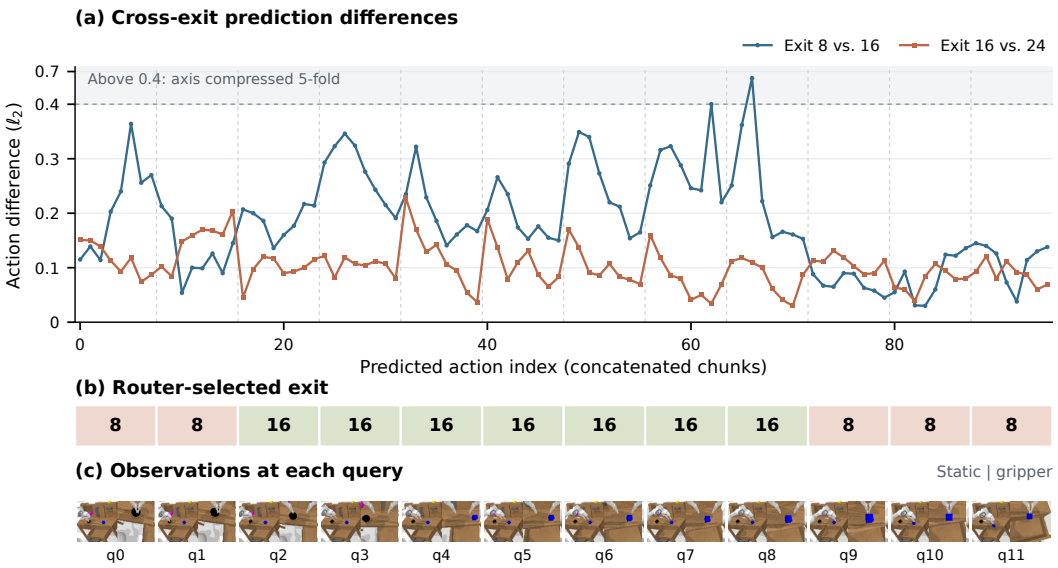

Figure 4: **Cross-exit predictions and routing along a CALVIN trajectory.** (a) Per-action ℓ_2 differences between adjacent exits; dashed lines separate eight-action prediction chunks. The vertical axis is linear up to 0.4 and compressed fivefold above 0.4, with ticks in the original units. (b) Recorded router recommendation at each query. (c) Static-camera and gripper-camera observations. Curves are approximately reconstructed from the supplied visualization, not raw measurements. For display, the exit-16/24 value at action index 11 is replaced by its immediate neighbors' mean; indices 63 and 66 retain neighbor-averaged values from the source figure.

sults support input-dependent depth selection, while showing that the cost of early exit varies across benchmarks.

Qualitative routing behavior. Figure 4 illustrates recorded router recommendations along a CALVIN trajectory. At the beginning (q_0–q_1), the router selects layer 8 despite appreciable disagreement between the layer-8 and layer-16 predictions. During the middle phase (q_2–q_8), their disagreement remains pronounced and the router selects layer 16. Toward the end (q_9–q_{11}), the predictions become more similar and the router returns to layer 8. One interpretation is that early motions tolerate a wider range of actions, whereas the middle phase requires more precise control. The displayed layer-16/24 differences are generally smaller during the middle phase, and the router never selects layer 24 in this example. This reconstructed diagnostic illustrates proposed exit choices, rather than verifying that they controlled the rollout; disagreement alone does not establish action accuracy or tolerance.

4.4 Ablation Studies

We examine joint exit supervision with robot-pretrained initialization (Table 4) and action-token design under adaptive inference with VLM-only initialization (Table 5). Comparisons within each table use matched training and evaluation settings.

Joint exit supervision. We supervise only the shallow or middle exit and evaluate it at the same depth as the jointly supervised model: layers 8/16 on CALVIN and 12/24 on RoboTwin. Joint supervision improves CALVIN AvgLen from 4.09 to 4.15 at the shallow exit and from 4.21 to 4.24 at the middle exit. On RoboTwin, the corresponding scores decrease slightly, from 62.1% to 61.5% and from 69.0% to 68.5%. Joint training therefore largely preserves individual-exit performance while supporting all three exits in one model, rather than improving every exit uniformly.

Specialized action tokens. The shared-token variant lets all three exits read a single, continuously updated token stream. Replacing this stream with independent exit streams, even without collaborative attention, increases adaptive CALVIN AvgLen from 4.12 to 4.23 and RoboTwin success from

Table 4: **Single-exit versus joint supervision.** Fixed-exit performance with robot-pretrained initialization; – denotes unsupervised exits that are not evaluated.

	CALVIN AvgLen ↑			RoboTwin SR (%) ↑		
Variant	Layer 8	Layer 16	Layer 24	Layer 12	Layer 24	Layer 36
Full SceVLA	4.15	4.24	4.39	61.5	68.5	82.6
Shallow-only supervision	4.09	–	–	62.1	–	–
Middle-only supervision	–	4.21	–	–	69.0	–

Table 5: **Token and attention ablations.** Adaptive performance with VLM-only initialization. Each variant trains its own router with the action policy frozen; latency is the mean batch-one policy-query time on an RTX 4090.

	CALVIN		RoboTwin 2.0	
Adaptive variant	AvgLen ↑	Latency (ms) ↓	SR (%) ↑	Latency (ms) ↓
SceVLA (ours)	4.36	68.9	72.7	79.6
Shared token stream	4.12	70.4	68.9	78.3
No collaborative attention	4.23	78.1	70.2	90.0

68.9% to 70.2%, although latency also increases. The full model further raises these scores to 4.36 and 72.7%, with query latency close to that of the shared-token variant (68.9 versus 70.4 ms on CALVIN; 79.6 versus 78.3 ms on RoboTwin). These comparisons support exit-specific representations; separate streams also introduce separate decoder parameters, so the comparison does not isolate token parameters alone.

Collaborative attention. Removing cross-exit attention while retaining independent streams reduces CALVIN AvgLen from 4.36 to 4.23 and RoboTwin success from 72.7% to 70.2%. It also increases mean latency from 68.9ms to 78.1 ms and from 79.6ms to 90.0 ms, respectively. With a separately trained router for each frozen policy, collaborative attention thus improves the observed adaptive accuracy–latency trade-off on both benchmarks.

5 Conclusion

We presented SceVLA, a dynamic-depth vision-language-action framework that combines sparse exits with specialized action tokens. Each exit develops a dedicated action representation, while directional collaborative attention enables deeper exits to reuse shallower-stream features without sending gradients back through the transferred keys and values. A separately trained routing token reads backbone and exit-stream representations to decide when further computation is unnecessary. Together, these components make intermediate prediction quality, cross-exit information sharing, and input-dependent stopping explicit parts of the policy design. SceVLA provides a framework for studying how action specialization and adaptive computation can support efficient robotic control. Future work can extend this design toward learned exit placement and routing objectives that account directly for closed-loop task outcomes.

Limitations

Our evaluation is limited to the CALVIN and RoboTwin 2.0 simulation benchmarks; real-robot experiments are needed to assess whether the observed accuracy–latency trade-offs carry over to physical systems with sensing noise and execution delays. In addition, SceVLA reduces computation in the language backbone and action prediction pathway but does not accelerate the visual encoder, which runs for every policy query. This fixed visual-processing cost limits the achievable end-to-end speedup and motivates combining adaptive depth with visual-encoding acceleration.

REFERENCES

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 Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Robotics: Science and Systems XXI*, RSS2025. Robotics: Science and Systems Foundation, June 2025. doi: 10.15607/rss.2025.xxi.010. URL http://dx.doi.org/10.15607/RSS.2025.XXI.010.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *arXiv preprint arXiv:2212.06817*, 2022.

Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation, 2025a. URL https://arxiv.org/abs/2410.08001.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions, 2025b. URL https://arxiv.org/abs/2505.06111.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-Scale Dense Networks for Resource Efficient Image Classification, 2017. URL https://arxiv.org/abs/1703.09844.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

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. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. *arXiv preprint arXiv:2502.19645*, 2025.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators, 2024. URL https://arxiv.org/abs/2311.01378.

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, Tao Kong, Hanbo Zhang, and Huaping Liu. What matters in building vision-language-action models for generalist robots, 2026. URL https://arxiv.org/abs/2412.14058.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation, 2025. URL https://arxiv.org/abs/2410.07864.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. *IEEE Robotics and Automation Letters (RA-L)*, 7(3):7327–7334, 2022.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Robotics: Science and Systems XXI*, RSS2025. Robotics: Science and Systems Foundation, June 2025. doi: 10.15607/rss.2025.xxi.012. URL http://dx.doi.org/10.15607/RSS.2025.XXI.012.

Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning, 2024. URL https://arxiv.org/abs/2412.12953.

Boyang Shen, Kaixiang Yang, Hao Wang, Qiuyu Yu, Qiang Xie, Qiang Li, and Zhiwei Wang. LoopVLA: Learning Sufficiency in Recurrent Refinement for Vision-Language-Action Models, 2026. URL https://arxiv.org/abs/2605.09948.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Yuxin Huang, Han Zhao, Donglin Wang, and Haoang Li. CEED-VLA: Consistency Vision-Language-Action Model with Early-Exit Decoding. *arXiv preprint arXiv:2506.13725*, 2025.

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing, 2026. URL https://arxiv.org/abs/2604.05014.

Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469. IEEE, December 2016. doi: 10.1109/icpr.2016.7900006. URL http://dx.doi.org/10.1109/ICPR.2016.7900006.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation, 2024. URL https://arxiv.org/abs/2412.15109.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. VLA-Adapter: An Effective Paradigm for Tiny-Scale Vision-Language-Action Model. *arXiv preprint arXiv:2509.09372*, 2025.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation, 2023. URL https://arxiv.org/abs/2312.13139.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient Vision-Language-Action Manipulation via Adaptive Token Caching. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 164448–164473. Curran Associates, Inc., 2025. doi: 10.52202/085713-5484. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f062da1973ac9ac61fc6d44dd7fa309f-Paper-Conference.pdf.

Senqiao Yang, Chengyao Wang, Yuxin Chen, Zixuan Wang, Longxiang Tang, Haokun Gui, Jinhui Ye, Changsheng Lu, Xiaoyang Wu, Mingkang Zhu, Pengguang Chen, Shu Liu, Zhuotao Tian, Hengshuang Zhao, Bei Yu, and Jiaya Jia. Beyond data scaling: Representation-centric continued

pre-training for vision-language-action models, 2026a. URL https://arxiv.org/abs/2608.27550.

Zebin Yang, Yijiahao Qi, Tong Xie, Bo Yu, Shaoshan Liu, and Meng Li. DySL-VLA: Efficient Vision-Language-Action Model Inference via Dynamic-Static Layer-Skipping for Robot Manipulation, 2026b. URL https://arxiv.org/abs/2602.22896.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic Inference of Multimodal Large Language Models for Efficient Robot Execution. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 56619–56643. Curran Associates, Inc., 2024. doi: 10.52202/079017-1803. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/67b0e7c7c2a5780aeefe3b79caac106e-Paper-Conference.pdf.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations, 2024. URL https://arxiv.org/abs/2403.03954.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. MoLe-VLA: Dynamic Layer-skipping Vision Language Action Model via Mixture-of-Layers for Efficient Robot Manipulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18764–18772, March 2026. ISSN 2159-5399. doi: 10.1609/aaai.v40i22.38945. URL http://dx.doi.org/10.1609/aaai.v40i22.38945.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

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 Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.