FLEXIBLE ACTION-ORIENTED IMAGE TOKENIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models commonly process hundreds of patch-level visual features at every control step, creating long visual contexts. These visual contexts constitute the largest computational bottleneck in VLAs. Existing VLA compression methods may discard control-relevant information at low token budgets or learn representations tied to a predetermined token count, limiting inference-time adaptation. We introduce Flexi-Act, a visual tokenizer that compresses patch features into a compact, ordered sequence of continuous tokens. This sequence preserves control-relevant information under aggressive compression and supports different visual-token budgets through its prefixes. To learn this ordering, we pretrain the tokenizer with nested dropout and a temporary reconstruction decoder that reconstructs frozen visual-encoder features from sampled prefixes, encouraging earlier tokens to capture broadly useful information while later tokens specialize. We then integrate the pretrained tokenizer into a downstream VLA and jointly fine-tune the tokenizer and policy across prefix lengths, allowing the same policy to switch between supported budgets at inference without retraining. At 16 tokens per camera view, Flexi-Act maintains strong control performance across LIBERO, LIBERO-Plus, and SIMPLER while using $\mathbf{16}\times$ fewer visual tokens and $\mathbf{4.5}\times$–$\mathbf{5.8}\times$ lower estimated FLOPs than OpenVLA-OFT. On LIBERO and LIBERO-Plus, it maintains near-lossless performance using only **1 visual token per camera view**.

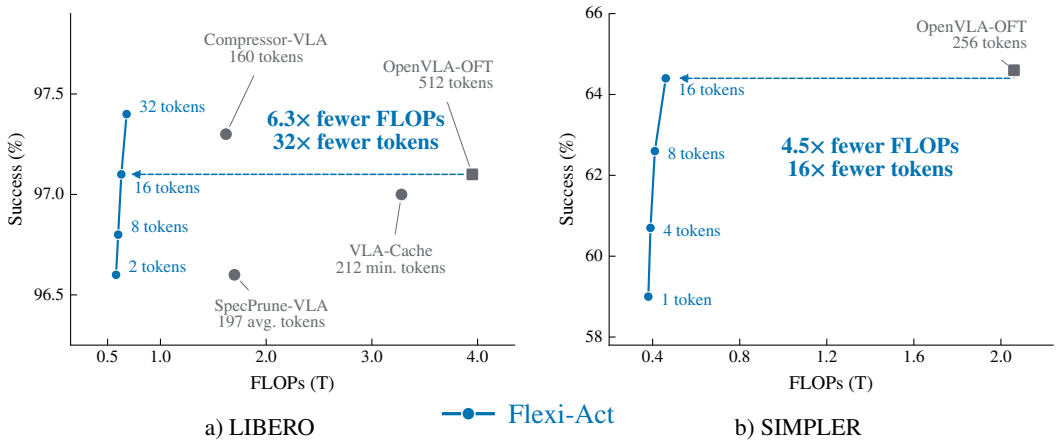
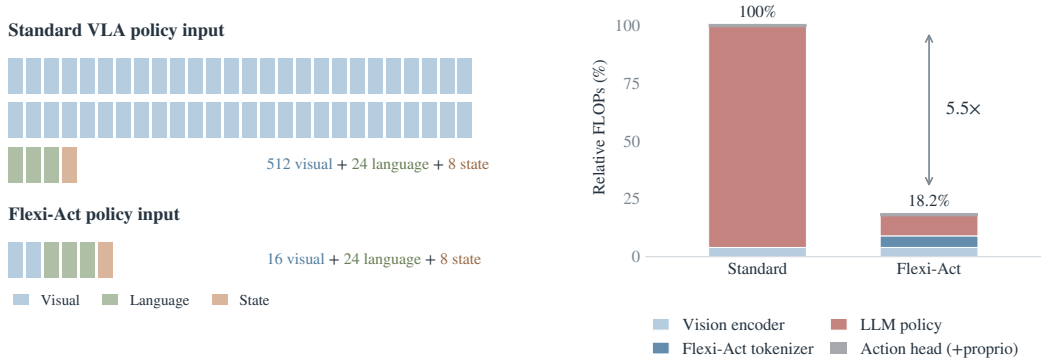


Figure 1: Success versus FLOPs on a) LIBERO and b) SIMPLER. Blue curves show Flexi-Act at different inference-time token budgets; gray markers show baselines. Point labels report total visual tokens.

1 INTRODUCTION

Vision-language-action (VLA) models offer a promising path toward general-purpose robot control (Brohan et al., 2023; Kim et al., 2024; Li et al., 2024a; NVIDIA et al., 2025; Team et al., 2025; Intelligence et al., 2025; Shukor et al., 2025). VLAs are typically fine-tuned from pretrained vision-language models on robot demonstrations to predict action sequences. They consequently inherit

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.



(a) Policy inputs

(b) Relative computation

Figure 2: Illustrative comparison of a standard VLA and Flexi-Act. **(a)** Flexi-Act compresses the visual tokens that dominate the policy input while retaining the language and state tokens. **(b)** Component-wise FLOPs, normalized to the standard VLA, illustrate how a shorter visual sequence reduces policy computation.

powerful visual encoders trained on large-scale image data. These encoders provide the rich semantic and spatial information required for control, but commonly represent each camera view as a dense sequence comprising hundreds of patch-level feature vectors. Multiple input views expand this visual context further. Processing such long sequences at every control step increases computational cost and inference latency, a particularly important concern for VLA deployment because real-world, high-frequency robot control requires rapid action generation. Consider the methods in Fig. 1: Most of their computational budget is consumed by the repeated processing of visual tokens. This is further illustrated in Fig. 2: In standard VLA policies over 90% of tokens are visual, and much of the policy computation is dominated by the processing of these tokens.

We introduce **Flexi-Act**, a highly-efficient ordered soft visual tokenizer designed to provide strong control performance at low and variable token budgets. Flexi-Act inserts a bidirectional-transformer soft tokenizer between the pretrained visual encoder and the policy backbone. For each camera view, learned register-tokens aggregate the full patch-level feature sequence into a compact, ordered sequence of continuous embeddings. We train Flexi-Act in two phases. During tokenizer pretraining, we freeze the visual encoder, sample prefixes of different lengths, and train a temporary reconstruction decoder to reconstruct the original patch-level features from each prefix. This objective encourages earlier tokens to retain information useful across many visual-token budgets, while later tokens contribute additional visual detail. We then discard the reconstruction decoder, integrate the pretrained tokenizer into a downstream VLA, and fine-tune the resulting policy on robot demonstrations while sampling different prefix lengths. By exposing the policy to multiple visual-token budgets during fine-tuning, we train a single model that can operate across all supported lengths. At inference, the policy receives only the selected prefix from each camera view, allowing it to switch seamlessly between visual-token budgets without retraining.

Flexi-Act significantly reduces the inference cost, see Fig. 2. Comprehensive experiments on LIBERO, LIBERO-Plus, and SIMPLER demonstrate that Flexi-Act retains strong control performance under aggressive visual-token compression. Using only one visual token per camera view, Flexi-Act retains 99.5% and 98.5% relative performance on LIBERO and LIBERO-Plus compared to the base-model. At the same time, Flexi-Act reduces visual tokens by up to $256\times$ and FLOPs by up to $6.8\times$. On SIMPLER, 16 Flexi-Act-tokens reach 99.7% of the base model's performance, with a $4.5\times$ reduction in FLOPs. These results show that Flexi-Act combines strong performance at low token budgets with the flexibility to allocate additional visual computation when beneficial.

2 Related Work

Vision-Language-Action Models extend vision-language modeling to embodied control by mapping visual observations and language instructions to robot actions. Early work established scalable

robot policies (Brohan et al., 2022) and transferred vision-language knowledge to embodied reasoning and action prediction (Driess et al., 2023; Brohan et al., 2023), while subsequent generalist models expanded the range of supported tasks, environments, and robot embodiments (Octo Model Team et al., 2024; Kim et al., 2024). More recent systems further scale pretrained vision-language backbones across diverse tasks and robot embodiments (Team et al., 2025). Their action-generation mechanisms span autoregressive token prediction (Brohan et al., 2023; Kim et al., 2024), direct continuous regression (Kim et al., 2025), diffusion-based generation (Li et al., 2024a; NVIDIA et al., 2025), and flow matching (Black et al., 2026; Intelligence et al., 2025; Shukor et al., 2025; Wu et al., 2026; Wang et al., 2026b). Although these models differ in architecture and action parameterization, they commonly rely on powerful visual encoders whose patch-level features form a dense visual sequence processed at every control step.

Visual-Token Compression for Vision-Language-Action Models. Work on efficient visual processing first studied dynamic token pruning and sparsification (Rao et al., 2021; Chen et al., 2024; Zhang et al., 2025) and token merging (Bolya et al., 2023) as ways to reduce the dense patch sequences consumed by vision and vision-language models. Recent VLA-specific methods adapt these ideas to the spatial and action-relevant requirements of robot control. Some methods exploit temporal redundancy through adaptive token caching and action reuse (Xu et al., 2025; Tan et al., 2025), combine token pruning with model scheduling or broader training-free acceleration (Li et al., 2025b; Yang et al., 2026), or learn pruning through differentiable selection (Jiang et al., 2025), self-speculative action-aware signals (Wang et al., 2026a), semantic-action alignment (Liu et al., 2026b), and dynamic action cues (Pei et al., 2025). Training-free approaches further compress visual features through token expansion and merging (Ye et al., 2026) or depth-guided token merging (Li et al., 2026a). Learned compression approaches use instructions to guide visual-token compression, routing, and sparsification (Gao et al., 2025; Li et al., 2025a), construct compact object-relation representations (Hanyu et al., 2026), or learn task-aware spatial resampling (Feng et al., 2026) for downstream action prediction. Recent methods improve pruning using interaction cues (Cheng et al., 2026), future-layer attention (Ma et al., 2026), or each token's effect on predicted actions (Li et al., 2026b). Together, these studies show that VLAs can operate with compact visual inputs. We instead learn a continuous, ordered representation whose prefixes support different token budgets within a single policy.

Ordered and Variable-Length Representations. Nested dropout orders learned features through stochastic truncation (Rippel et al., 2014), while Matryoshka representation learning explicitly trains prefixes to remain useful across representation budgets (Kusupati et al., 2022). Multimodal extensions apply this principle to visual features and learned query sequences, allowing one model to process different numbers of visual embeddings at inference (Cai et al., 2025; Hu et al., 2024; Kuzucu et al., 2026). Compact visual tokenizers replace two-dimensional feature grids with one-dimensional latent sequences (Yu et al., 2024), while flexible variants support variable-length tokenization for images and videos through truncatable one-dimensional sequences (Bachmann et al., 2025; Miwa et al., 2025; Yan et al., 2025; Atanov et al., 2026). Related work studies recurrent token allocation, the distribution of information across token positions, and adaptation of sequence length to image complexity, including discrete representations learned through pretrained visual-feature supervision (Duggal et al., 2024; Fu et al., 2026; Chen et al., 2026; Wyrwiński et al., 2026). These visual representations are primarily evaluated through visual understanding, reconstruction, or generation. In robotics, prefix-based representation learning has also been applied to action tokenization, enabling ordered discrete prefixes to reconstruct complete action chunks (Liu et al., 2026a). Flexi-Act applies prefix ordering to visual observations, learning continuous embeddings in the feature space of a pretrained VLA visual encoder and training one downstream policy across prefix lengths for closed-loop robot control.

3 PRELIMINARIES

Vision-Language-Action Models. At timestep t, a vision-language-action (VLA) model π_θ receives a visual observation $\mathbf{o}_t$, a language instruction ℓ, and, when available, the robot's proprioceptive state $\mathbf{s}_t$. The policy predicts an action chunk according to

$$\widehat{\mathbf{A}}_t = \pi_\theta(\mathbf{o}_t, \ell, \mathbf{s}_t)\,, \tag{1}$$

where $\widehat{\mathbf{A}}_t \in \mathbb{R}^{H \times d_a}$ contains H future actions of dimension d_a. Let $\mathbf{A}_t \in \mathbb{R}^{H \times d_a}$ denote the corresponding ground-truth action chunk. A typical VLA comprises a visual encoder that extracts features from the camera observations, a projector that maps these features into the policy's embedding space, a multimodal policy backbone, and an action head that produces the action chunk.

Downstream supervised fine-tuning adapts the VLA to a target task, environment, or robot embodiment using a demonstration dataset $\mathcal{D}$. It minimizes an action-prediction objective,

$$\mathcal{L}_{\mathrm{VLA}}(\theta) = \mathbb{E}_{(\mathbf{o}_t, \ell, \mathbf{s}_t, \mathbf{A}_t) \sim \mathcal{D}} \left[\mathcal{L}_{\mathrm{act}}(\pi_\theta(\mathbf{o}_t, \ell, \mathbf{s}_t), \mathbf{A}_t) \right], \tag{2}$$

where $\mathcal{L}_{\mathrm{act}}$ depends on the model's action representation and prediction strategy.

At inference time, many VLAs generate actions autoregressively or semi-autoregressively using a transformer-based policy backbone. Even when the action head predicts multiple actions in parallel, the backbone processes the complete multimodal input sequence before producing an action chunk. Its inference cost therefore grows with the input sequence length: token-wise projections and feed-forward layers scale linearly with the number of input tokens, while self-attention scales quadratically. Flexi-Act speeds up VLA generation by shortening the visual context.

Visual Representations in VLAs. The multimodal context processed by a VLA contains language, state, and visual embeddings, with visual embeddings typically accounting for the largest portion of the input sequence (Kim et al., 2024; 2025; Black et al., 2026; Intelligence et al., 2025). Vision-language models obtain these embeddings by representing each image as a dense sequence of patch-level features. For an observation containing images from C camera views, $\mathbf{o}_t = \{I_t^{(c)}\}_{c=1}^{C}$, the visual encoder processes each view as

$$\mathbf{X}_t^{(c)} = E_{\mathrm{vis}}\left(I_t^{(c)}\right) = \left[\mathbf{x}_{t,1}^{(c)}, \ldots, \mathbf{x}_{t,N}^{(c)}\right] \in \mathbb{R}^{N \times d_v}, \tag{3}$$

where N denotes the number of features in the encoder's fixed patch grid and d_v denotes their dimensionality. Standard VLA interfaces project and flatten every grid location into a separate policy input, producing a visual sequence whose length grows with the area of the patch grid. The policy backbone processes this dense visual sequence at every control step, making the fixed patch grid a central bottleneck for VLA inference. Flexi-Act uses an ordered variable-length representation to summarize the grid as a compact, one-dimensional sequence.

Ordered Variable-Length Representations. Following ordered representation learning and flexible-length visual tokenization (Rippel et al., 2014; Bachmann et al., 2025), we define an ordered variable-length tokenizer as producing a sequence of $K_{\max}$ continuous tokens and exposing shorter representations through prefixes of this sequence. Let $\mathcal{K} \subseteq \{1, \ldots, K_{\max}\}$ denote the supported token budgets and let $p_{\mathcal{K}}$ denote their training-time sampling distribution.

$$\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{K_{\max}}], \qquad \mathbf{Z}^{(K)} = \mathrm{Prefix}_K(\mathbf{Z}) = [\mathbf{z}_1, \ldots, \mathbf{z}_K], \tag{4}$$

where $\mathbf{Z}^{(K)}$ denotes the representation under a supported token budget $K \in \mathcal{K}$. Unlike patch-level features whose positions correspond to locations in a fixed spatial grid, the positions in $\mathbf{Z}$ are ordered by their usefulness across token budgets. Each prefix therefore represents the complete visual input at a different capacity, with longer prefixes providing additional information.

Nested-dropout provides a mechanism for learning this ordering (Rippel et al., 2014) and has been applied to variable-length visual tokenization for image generation (Bachmann et al., 2025). During training, nested-dropout samples a token budget $K \sim p_{\mathcal{K}}$ and optimizes the model using only the corresponding prefix $\mathbf{Z}^{(K)}$. A token $\mathbf{z}_j$ is retained whenever $K \geq j$, exposing earlier positions to more training budgets than later positions. This encourages earlier tokens to capture information useful to all representation lengths, while later tokens add detail when larger budgets are available. Flexi-Act uses nested-dropout to build a variable-length, action-oriented soft tokenizer.

4 METHOD

For each input view, Flexi-Act replaces the fixed patch grid with a compact, ordered sequence of continuous visual tokens. A bidirectional tokenizer uses learned register tokens to aggregate information from the complete patch-level feature sequence into this one-dimensional representation

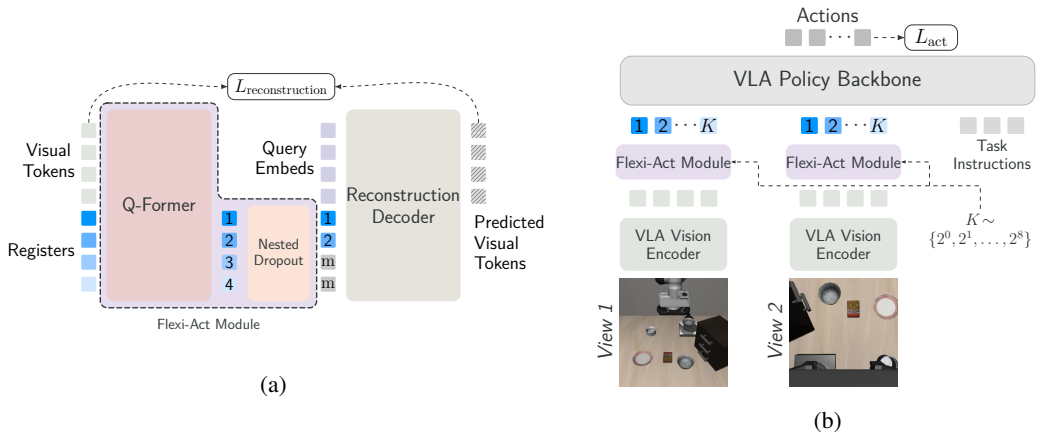


Figure 3: **Two-stage Flexi-Act training. (a) Stage 1: tokenizer pretraining.** The tokenizer encoder learns an ordered sequence of continuous visual tokens. Register tokens first aggregate information from the visual features, after which nested-dropout retains a variable-length prefix—the first K tokens out of $K_{\max}$—and a reconstruction decoder trains this prefix to reconstruct the complete visual feature set. **(b) Stage 2: VLA fine-tuning.** The pretrained tokenizer represents each observation with a sampled prefix of length K, and the VLA policy learns to predict actions across all supported budgets.

(Section 4.1). To make prefixes of different lengths independently useful, we pretrain the tokenizer with nested dropout and require each sampled prefix to reconstruct the complete features produced by the frozen visual encoder (Section 4.2). We then discard the reconstruction decoder, integrate the pretrained tokenizer into a VLA, and jointly fine-tune the tokenizer and policy while sampling different prefix lengths, teaching a single policy to predict actions across all supported token budgets (Section 4.3). At inference, we retain only the selected prefix from each camera view, allowing the same policy to adjust its visual computation by changing the token budget without retraining (Section 4.4). The result is a single flexible tokenizer that outperforms prior methods across a range of accuracy and compression levels.

4.1 Flexi-Act Architecture

Flexi-Act transforms the visual encoder's patch-level features into a compact, learned representation that serves as the VLA policy's visual input. For camera view c at timestep t, we begin with the patch-level features $\mathbf{X}_t^{(c)} \in \mathbb{R}^{N \times d_v}$ of the visual encoder in Eq. (3). We use a Q-Former Q_ϕ to project the patch-level features $\mathbf{X}_t^{(c)}$ onto a fixed set of $K_{\max}$ learned register embeddings $\mathbf{R}_\phi \in \mathbb{R}^{K_{\max} \times d_z}$. This produces an over-complete set of soft visual tokens:

$$\mathbf{Z}_t^{(c)} = Q_\phi\left(\mathbf{X}_t^{(c)}, \mathbf{R}_\phi\right) = \left[\mathbf{z}_{t,1}^{(c)}, \ldots, \mathbf{z}_{t,K_{\max}}^{(c)}\right]. \tag{5}$$

The parameters include projection, positional embedding and transformer components of the Q-Former Q_ϕ, and the registers $\mathbf{R}_\phi$. We discard the transformer outputs corresponding to $\mathbf{X}_t^{(c)}$ and use the register outputs as a compact bottleneck over the original grid. We apply the same tokenizer independently to every camera view. The module closely mimics the perceiver architecture (Alayrac et al., 2022).

4.2 Learning Ordered Visual Embeddings

The tokenizer architecture does not assign predefined importance to register positions. We learn their ordering through nested-dropout pretraining, encouraging shorter prefixes to preserve broadly useful visual information and longer prefixes to add further detail.

Following the nested-dropout formulation in Eq. (4), we sample a budget $K \sim p_{\mathcal{K}}$ and train the prefix $\mathbf{Z}_t^{(c,K)} = \text{Prefix}_K(\mathbf{Z}_t^{(c)})$ to reconstruct the complete patch-level representation $\mathbf{X}_t^{(c)}$, as illus-

trated in Fig. 3a. Every supported prefix therefore shares the same reconstruction target. To batch prefixes of different lengths, we replace positions after K with a learned mask embedding $\mathbf{m} \in \mathbb{R}^{d_z}$, yielding $\widetilde{\mathbf{Z}}_t^{(c,K)} = [\mathbf{z}_{t,1}^{(c)}, \ldots, \mathbf{z}_{t,K}^{(c)}, \mathbf{m}, \ldots, \mathbf{m}]$. We use the mask embedding only during pretraining.

We use ψ to denote all training-only reconstruction parameters, including the mask embedding and the reconstruction decoder's weights, learned patch queries, and positional embeddings. A training-only reconstruction decoder D_ψ reconstructs the complete patch-level feature sequence from the retained prefix. The reconstruction decoder uses one query for each target patch location, adds learned positional embeddings $\mathbf{P}_{\text{dec}}$ to the queries, applies self-attention among them, and cross-attends to $\widetilde{\mathbf{Z}}_t^{(c,K)}$. It produces the reconstruction $\widehat{\mathbf{X}}_t^{(c,K)} = D_\psi(\widetilde{\mathbf{Z}}_t^{(c,K)}) \in \mathbb{R}^{N \times d_v}$. Let $\mathbf{x}_n$ and $\widehat{\mathbf{x}}_n$ denote the target and reconstructed features at patch location n, respectively. We train the tokenizer and reconstruction decoder using the cosine reconstruction loss, $\mathcal{L}_{\text{rec}} = \frac{1}{N} \sum_{n=1}^{N} \left(1 - \frac{\widehat{\mathbf{x}}_n^\top \mathbf{x}_n}{\|\widehat{\mathbf{x}}_n\|_2 \|\mathbf{x}_n\|_2}\right)$, averaged over training images and sampled token budgets.

Reconstructing the pretrained visual features rather than image pixels focuses the tokenizer on the semantic and spatial information already captured by the visual encoder. After pretraining, we discard the reconstruction decoder and mask embedding and retain only the ordered Flexi-Act tokenizer. Section 4.3 next adapts these visual prefixes for downstream action prediction.

4.3 Multi-Budget VLA Fine-Tuning

Stage 1 organizes visual information into ordered prefixes, but the downstream policy must still learn to predict actions from each supported prefix length. As illustrated in Fig. 3b, we insert the pretrained Flexi-Act tokenizer between the visual encoder and the VLA policy backbone and apply it independently to the patch features from each camera view. During fine-tuning, we sample a token budget $K \sim p_{\mathcal{K}}$ for each training example and use the same budget across all views. We concatenate the prefixes from all C camera views at the shared budget K as $\bar{\mathbf{Z}}_t^{(K)} = \left[\mathbf{Z}_t^{(1,K)}, \mathbf{Z}_t^{(2,K)}, \ldots, \mathbf{Z}_t^{(C,K)}\right] \in \mathbb{R}^{CK \times d_z}$, yielding CK visual tokens ordered within each view. A learned projector P_ω maps $\bar{\mathbf{Z}}_t^{(K)}$ into the input space of the policy backbone, where it replaces the original patch-level visual sequence; ω denotes its parameters. The policy combines these projected visual embeddings with the language instruction and robot state to predict an action chunk.

We optimize the resulting VLA using the supervised action-prediction objective in Eq. 2. Sampling different values of K exposes the same policy to multiple visual-representation lengths and teaches it to act from both compact prefixes and the additional detail available at larger budgets. This multi-budget fine-tuning allows one policy to support every prefix length learned during Stage 1.

4.4 Flexible-Budget Inference

At inference, we select a token budget $K \in \mathcal{K}$ according to the computational constraints of the deployment setting. For each camera view, Flexi-Act retains the first K ordered tokens, so an observation with C views contributes CK visual tokens instead of the CN tokens produced by the original patch grids. Smaller values of K reduce the cost of processing visual context, while larger values provide the policy with progressively more visual information. Because multi-budget fine-tuning trains a single policy across all supported prefix lengths, the same policy can seamlessly switch between budgets by changing K without retraining.

5 Experiments

Benchmarks. We evaluate Flexi-Act on LIBERO, LIBERO-Plus, and SIMPLER. LIBERO (Liu et al., 2023) is a simulated benchmark for language-conditioned robot manipulation. We evaluate on its four standard ten-task suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, which test manipulation across variations in spatial relationships, manipulated objects, task goals, and long-horizon behavior, respectively. We additionally evaluate on LIBERO-Plus (Fei et al., 2025), which extends LIBERO with controlled perturbations to object layouts, camera viewpoints, robot initial states, language instructions, lighting conditions, background textures, and sensor ob-

Table 1: Success rates (%) on the LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long task suites. Results for the efficient VLA baselines are taken from Compressor-VLA (Gao et al., 2025). The best and second-best success rates in each column, excluding OpenVLA-OFT, are shown in **bold** and underlined, respectively.

Model	Spatial	Object	Goal	Long	Avg.	TFLOPs	Tokens
OpenVLA-OFT (Kim et al., 2025)	97.6	98.4	97.9	94.5	97.1	3.95	512
Efficient VLA Methods							
SP-VLA (Li et al., 2025b)	75.4	85.6	84.4	54.2	74.9	3.10	229
FastV (Chen et al., 2024)	96.8	81.0	96.4	73.0	86.8	3.18	256
SparseVLM (Zhang et al., 2025)	96.8	94.2	97.6	93.6	95.6	3.04	256
SpecPrune-VLA (Wang et al., 2026a)	98.2	96.3	97.7	94.0	96.6	1.70	197
VLA-Cache (Xu et al., 2025)	**99.0**	97.7	97.4	93.6	97.0	3.28	212+
Compressor-VLA (Gao et al., 2025)	<u>98.8</u>	**99.2**	96.4	**94.8**	<u>97.3</u>	1.62	160
Flexi-Act ($K = 1$)	97.4	98.5	97.5	93.1	96.6	**0.58**	**2**
Flexi-Act ($K = 4$)	97.6	98.4	97.5	93.6	96.8	0.60	8
Flexi-Act ($K = 8$)	97.5	98.8	<u>97.9</u>	94.0	97.1	0.63	16
Flexi-Act ($K = 16$)	98.0	<u>99.0</u>	**98.2**	<u>94.2</u>	**97.4**	0.68	32

servations. Finally, we use SIMPLER (Li et al., 2024b), a simulated evaluation framework designed to reproduce the visual and control characteristics of common real-robot settings. We evaluate the Google Robot *Pick Coke Can*, *Move Near*, and *Open/Close Drawer* tasks under the Visual Matching protocol. We report the average rollout success rate for all benchmarks.

Base models. We evaluate Flexi-Act with two VLA backbones that use distinct action-generation formulations: OpenVLA-OFT (Kim et al., 2025) and π_0 (Black et al., 2026). OpenVLA-OFT builds on the 7B-parameter OpenVLA model, which combines a fused SigLIP and DINOv2 visual encoder with a Llama-2 language-model backbone. Its optimized fine-tuning recipe replaces autoregressive discrete-action prediction with parallel action chunking and a continuous action head trained using an ℓ_1 regression objective. In contrast, π_0 is a 3.3B-parameter flow-matching VLA initialized from the PaliGemma vision-language model. It augments the 3B-parameter VLM backbone with a 300M-parameter action expert that generates continuous action chunks through flow matching. Together, these models allow us to evaluate Flexi-Act across both regression-based and flow-matching VLA architectures.

Stage 1 tokenizer pretraining. For Stage 1, we pretrain the visual tokenizer using images drawn from the Open X-Embodiment dataset (Collaboration et al., 2025). We use the 27 datasets included in OpenVLA's `oxe_magic_soup_plus` collection and extract the primary third-person camera view from each episode. We sample up to ten frames distributed across each episode and resize them to 224×224, yielding approximately 12.3 million feature sequences. We construct a deterministic episode-level split that reserves 5% of the episodes for validation, ensuring that frames from the same episode do not appear in both splits. We provide the complete dataset-curation and feature-caching procedure in Section A.

Stage 2 policy fine-tuning. For LIBERO, we fine-tune a separate policy on the training demonstrations from each of the four standard task suites. Our results in Tables 1 and 2 therefore use the checkpoint trained for the corresponding evaluation suite. For LIBERO-Plus, we first fine-tune each backbone on a mixture of training demonstrations from all four standard LIBERO suites. We then continue fine-tuning the resulting checkpoint on a mixed LIBERO-Plus training set and evaluate it across the seven perturbation dimensions reported in Table 3. For SIMPLER, we fine-tune on the Fractal dataset (Brohan et al., 2022) before evaluating the resulting policy on the Google Robot tasks.

Baselines. We compare Flexi-Act with both its corresponding base models and prior efficient VLA methods. For a fair comparison, we fine-tune each base model using the same Stage 2 training data and fine-tuning protocol as its Flexi-Act counterpart. On LIBERO, we additionally compare

the OpenVLA-OFT instantiation against the efficient VLA baselines SP-VLA, FastV, SparseVLM, SpecPrune-VLA, VLA-Cache, and Compressor-VLA. We take the success rates, token counts, and FLOP estimates for these prior methods from the comparison reported by Compressor-VLA. Across LIBERO, LIBERO-Plus, and SIMPLER, we evaluate Flexi-Act at multiple token budgets against the correspondingly fine-tuned OpenPI and OpenVLA-OFT base models.

Implementation details. For Stage 1, we train a separate Flexi-Act tokenizer for each VLA backbone using cached features from its frozen visual encoder. We optimize the tokenizer and reconstruction decoder with AdamW using a global batch size of 2048, learning rates of 3×10^{-4} and 10^{-3}, respectively, and a cosine schedule with 1,000 warmup steps. We initialize downstream fine-tuning from the 25,000-step checkpoint; Section B reports the full Stage 1 reconstruction trajectories. At both stages, we sample K uniformly from $\mathcal{K} = \{1, 2, 4, 8, 16, 32, 64, 128, 256\}$; for observations containing multiple camera views, we share the sampled budget across all views.

For Stage 2, we use backbone-specific fine-tuning recipes. For OpenVLA-OFT, we fully fine-tune the tokenizer and newly initialized policy components while applying rank-32 LoRA (Hu et al., 2021) to the visual encoder and language-model backbone. We use AdamW with a learning rate of 5×10^{-5} for the tokenizer and 5×10^{-4} for the remaining trainable parameters, a global batch size of 64, 2,500 warmup steps, and image augmentation.

For π_0, we jointly fine-tune the complete policy and tokenizer using AdamW with a global batch size of 32. On LIBERO, we use a cosine schedule with 1,000 warmup steps and a peak learning rate of 2.5×10^{-5}. For LIBERO-Plus and Fractal, we use a constant learning rate of 5×10^{-4}. We report results for both backbones at $K \in \{1, 4, 8, 16\}$. We compute TFLOPs using the `thop` library (Zhu, 2022). We provide complete architecture and training hyperparameters for both stages in Section E.

5.1 Main Results

Comparison with efficient VLA methods on LIBERO. On the standard LIBERO suites (Table 1), Flexi-Act maintains strong performance at substantially smaller token budgets. At its most compressed setting, $K = 1$, it achieves 96.6% average success using one visual token per camera view — two visual tokens in total, closely matching OpenVLA-OFT while reducing its token count by $256\times$ and its FLOPs by $6.8\times$. At $K = 16$, Flexi-Act achieves the highest average success rate of 97.4%, outperforming both OpenVLA-OFT and Compressor-VLA. It does so with $16\times$ fewer tokens and $5.8\times$ fewer FLOPs than OpenVLA-OFT, and $5\times$ fewer tokens and $2.4\times$ fewer FLOPs than Compressor-VLA. We observe a similar trend with OpenPI in Table 2, where Flexi-Act closely matches the uncompressed policy across token budgets while using as few as two visual tokens.

Table 2: OpenPI success rates (%) on the four LIBERO task suites across token budgets.

Model	Tokens	TFLOPs	Spatial	Object	Goal	Long	Avg.
OpenPI (π_0)	512	1.535	96.8	98.8	95.8	85.2	94.2
Flexi-Act	2	0.420	96.6	98.3	95.2	83.9	93.5
Flexi-Act	8	0.430	96.6	98.2	95.4	84.0	93.6
Flexi-Act	16	0.450	96.9	98.6	95.7	84.6	94.0
Flexi-Act	32	0.485	96.9	98.7	95.8	85.0	94.1

Results on LIBERO-Plus. We evaluate robustness to controlled visual and environmental variations on LIBERO-Plus (Table 3). Using only one token per camera view, Flexi-Act achieves 78.4% overall success, compared with 79.6% for OpenVLA-OFT base model using 256 visual tokens per camera view, while matching or outperforming it on five of the seven perturbation dimensions. In particular, Flexi-Act improves success under robot initial-state and sensor-noise perturbations from 30.3% to 38.6% and from 89.3% to 93.4%, respectively. Performance remains stable across token budgets and reaches 78.9% at $K = 16$, demonstrating graceful adaptation to different visual-compute budgets.

Results on SIMPLER. We further evaluate Flexi-Act on the SIMPLER Google Robot tasks (Table 4), which provide a distinct environment and robot embodiment from LIBERO. With OpenVLA-

Table 3: LIBERO-Plus success rates (%) across perturbations and token budgets.

Model	Tokens	TFLOPs	Cam.	Robot	Lang.	Light	BG	Noise	Layout	Total
OpenPI	512	1.535	54.5	18.2	79.1	90.2	85.5	89.9	77.6	69.2
Flexi-Act	2	0.420	48.1	21.3	74.1	88.8	84.1	92.2	74.1	67.4
Flexi-Act	8	0.430	48.5	21.9	75.2	88.6	84.2	92.8	74.9	68.0
Flexi-Act	16	0.450	49.4	22.5	75.9	89.7	85.0	93.3	75.8	68.7
Flexi-Act	32	0.485	50.1	23.3	76.1	90.1	85.1	93.9	76.3	69.2
OpenVLA-OFT	512	3.95	92.8	30.3	85.8	94.9	93.9	89.3	77.6	79.6
Flexi-Act	2	0.58	86.1	38.6	69.7	95.5	95.9	93.4	78.4	78.4
Flexi-Act	8	0.60	86.3	38.8	69.8	95.5	96.0	93.6	78.3	78.5
Flexi-Act	16	0.63	86.4	39.0	70.3	95.8	96.1	93.5	78.5	78.7
Flexi-Act	32	0.68	86.9	39.3	70.5	96.2	96.1	93.7	78.6	78.9

OFT, the average success rate increases consistently from 59.0% at $K = 1$ to 64.4% at $K = 16$, nearly recovering the base model's 64.6% performance. The steady improvement shows the benefit of increasing the token budget when additional visual context is useful. We observe the same trend with OpenPI, where performance improves with the token budget and approaches that of the corresponding base model.

Table 4: SIMPLER success rates (%) across tasks and token budgets.

Model	Tokens	TFLOPs	Pick Coke Can	Move Near	Open/Close Drawer	Avg.
OpenPI	256	0.870	62.2	59.0	44.1	56.0
Flexi-Act (π_0)	1	0.305	50.9	53.7	41.1	49.0
Flexi-Act (π_0)	4	0.310	54.3	55.4	42.1	51.2
Flexi-Act (π_0)	8	0.320	58.1	58.1	42.9	53.8
Flexi-Act (π_0)	16	0.340	61.9	58.9	44.3	55.9
OpenVLA-OFT	256	2.06	69.7	68.3	53.2	64.6
Flexi-Act (OFT)	1	0.38	61.3	63.4	51.1	59.0
Flexi-Act (OFT)	4	0.39	63.6	65.1	52.0	60.7
Flexi-Act (OFT)	8	0.41	66.2	66.5	52.7	62.6
Flexi-Act (OFT)	16	0.46	69.6	68.0	53.3	64.4

Effect of the token budget. On LIBERO and LIBERO-Plus, increasing the token budget produces only modest changes in success rate because a single token already recovers most of the base-model performance. On LIBERO, Flexi-Act achieves 96.6% at $K = 1$ with OpenVLA-OFT and 93.5% with OpenPI, compared with 97.1% and 94.2% for the respective base models. Similarly, on LIBERO-Plus, performance increases only from 78.4% at $K = 1$ to 78.9% at $K = 16$. SIMPLER exhibits a more marked benefit from additional tokens: success rises from 59.0% at $K = 1$ to 64.4% at $K = 16$, nearly recovering the base model's 64.6%. These results show that Flexi-Act can operate effectively with a minimal token budget while allowing additional visual capacity when it provides a measurable benefit. We report results at additional token budgets in Section C.

6 Conclusion

We introduced Flexi-Act, an ordered continuous visual tokenizer that replaces a fixed patch-level feature grid with a compact sequence whose prefixes support different visual-token budgets. We pretrain the tokenizer to reconstruct frozen visual features from sampled prefixes and then fine-tune the downstream VLA across prefix lengths, enabling a single policy to operate at multiple budgets without retraining. Experiments on LIBERO, LIBERO-Plus, and SIMPLER show that Flexi-Act preserves strong control performance at low token counts while degrading gracefully as the available visual context is reduced. These results demonstrate that compact, variable-length visual representations can reduce VLA inference cost without tying a policy to a single deployment budget.

AI USE STATEMENT

We used generative AI tools to assist with implementing methods, generating and reviewing code, supporting literature review, polishing portions of the paper, and creating or revising figures and tables. We did not use generative AI to generate synthetic datasets, develop theoretical models, formulate or prove mathematical claims, clean or reformat datasets, perform qualitative analysis, or translate text. The authors reviewed and verified all AI-assisted work, tested all AI-assisted code, and checked citations and technical claims against the original sources and experimental artifacts. We take full responsibility for the final content of this work, including all text, claims, code, figures, tables, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work focuses on improving the computational efficiency of vision-language-action models by learning compact representations of their visual inputs. Our experiments use existing datasets and simulated evaluation environments and do not involve human subjects or the collection of personal or sensitive data. We are not aware of ethical concerns specific to the proposed method beyond the general safety and reliability considerations associated with deploying learned robotic policies in real-world environments.

REPRODUCIBILITY STATEMENT

We plan to release the code and experimental artifacts required to reproduce the results presented in this paper. The Method and Experiments sections specify the model architecture, training procedure, datasets, baselines, and evaluation protocols, while the appendix provides the complete hyperparameters and additional implementation details. Together with the released code and configurations, these materials are intended to support reproduction of our experiments across all evaluated models, benchmarks, and token budgets.

REFERENCES

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob L Menick, Sebastian Borgeaud, Andy Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikoł aj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karén Simonyan. Flamingo: a visual language model for few-shot learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736. Curran Associates, Inc., 2022. doi: 10.52202/068431-1723. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/960a172bc7fbf0177ccccbb411a7d800-Paper-Conference.pdf.

Andrei Atanov, Jesse Allardice, Roman Bachmann, Oğuzhan Fatih Kar, R Devon Hjelm, David Griffiths, Peter Fu, Afshin Dehghan, and Amir Zamir. Videoflextok: Flexible-length coarse-to-fine video tokenization. In *Forty-third International Conference on Machine Learning*, 2026.

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025.

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, 2026. URL https://arxiv.org/abs/2410.24164.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

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.

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, Alex 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, Jaspiar 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. In *arXiv preprint arXiv:2307.15818*, 2023.

Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. *Proceedings of the International Conference on Learning Representation*, 2025.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024.

Zeyuan Chen, Kai Zhang, Zhuowen Tu, and Yuanjun Xiong. Soft tail-dropping for adaptive visual tokenization. In *ECCV*, 2026.

Jintao Cheng, Weibin Li, Haozhe Wang, Gang Wang, Yipu Zhang, Xiaoyu Tang, Jin Wu, Xieyuanli Chen, Yunhui Liu, and Wei Zhang. Training-free interaction-aligned visual token pruning for efficient embodied manipulation, 2026. URL https://arxiv.org/abs/2603.22991.

Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Andrey Kolobov, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Felipe Vieira Frujeri, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guangwen Yang, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homanga Bharadhwaj, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jay Vakil, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey

Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi "Jim" Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Muhammad Zubair Irshad, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R Sanketi, Patrick "Tree" Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shubham Tulsiani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vikash Kumar, Vincent Vanhoucke, Vitor Guizilini, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiangyu Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yansong Pang, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yongqiang Dou, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Fu, and Zipeng Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2025. URL https://arxiv.org/abs/2310.08864.

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. URL https://arxiv.org/abs/2303.03378.

Shivam Duggal, Phillip Isola, Antonio Torralba, and William T. Freeman. Adaptive length image tokenization via recurrent allocation, 2024. URL https://arxiv.org/abs/2411.02393.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models, 2025. URL https://arxiv.org/abs/2510.13626.

Yixu Feng, Zinan Zhao, Yanxiang Ma, Chenghao Xia, Chengbin Du, Yunke Wang, and Chang Xu. See what matters: Differentiable grid sample pruning for generalizable vision-language-action model. In *Forty-Third International Conference on Machine Learning (ICML)*, 2026.

Zixuan Fu, Lanqing Guo, Chong Wang, Binbin Song, Ding Liu, and Bihan Wen. Improving flexible image tokenizers for autoregressive image generation, 2026. URL https://arxiv.org/abs/2601.01535.

Juntao Gao, Feiyang Ye, Jing Zhang, and Wenjing Qian. Compressor-vla: Instruction-guided visual token compression for efficient robotic manipulation, 2025. URL https://arxiv.org/abs/2511.18950.

Taisei Hanyu, Nhat Chung, et al. Slotvla: Towards modeling of object-relation representations in robotic manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Wenbo Hu, Zi-Yi Dou, Liunian Harold Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Mqt-llava: Matryoshka query transformer for large vision-language models. In *The 38th Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

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.

Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning, 2025. URL https://arxiv.org/abs/2509.12594.

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. URL https://arxiv.org/abs/2406.09246.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, December 2022.

Selim Kuzucu, Alessio Tonioni, Vasile Lup, Bernt Schiele, Federico Tombari, and Muhammad Ferjad Naeem. Parcel: Pool-anchored resampling with conditioned elastic queries for efficient vision-language understanding, 2026. URL https://arxiv.org/abs/2605.30126.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024a. URL https://arxiv.org/abs/2411.19650.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action model via instruction-driven routing & sparsification. *Advances in neural information processing systems*, 2025a.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation, 2024b. URL https://arxiv.org/abs/2405.05941.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shutao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration, 2025b. URL https://arxiv.org/abs/2506.12723.

Yuquan Li, Lianjie Ma, Han Ding, and Lijun Zhu. Depthcache: Depth-guided training-free visual token merging for vision-language-action model inference, 2026a. URL https://arxiv.org/abs/2603.10469.

Zhuoyuan Li, Rui Zhao, Jin Wang, Hanwei Zhu, Cong Zhang, Giuseppe Valenzise, Weisi Lin, and Kin-Man Lam. Just noticeable difference modeling for token compression in vision-language-action models, 2026b. URL https://arxiv.org/abs/2608.21247.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

Chaoqi Liu, Yue Zhao, Haonan Chen, Xiaoshen Han, Jiawei Gao, Ehsan Adeli, and Yilun Du. Ordered action tokens for visuomotor policy learning, 2026a. URL https://arxiv.org/abs/2607.21670.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference, 2026b. URL https://arxiv.org/abs/2511.16449.

Shilin Ma, Chubin Zhang, Changyuan Wang, Yuji Wang, Yue Wu, Zixuan Wang, Jingqi Tian, Zheng Zhu, and Yansong Tang. Safe-pruner: Semantic attention-guided future-aware token pruning for efficient vision-language-action manipulation, 2026. URL https://arxiv.org/abs/2605.29662.

Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-d-piece: Image tokenizer meets quality-controllable compression, 2025. URL https://arxiv.org/abs/2501.10064.

NVIDIA, Johan Bjorck, Nikita Cherniadev Fernando Castañeda, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. In *ArXiv Preprint*, March 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation, 2025. URL https://arxiv.org/abs/2509.22093.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Oren Rippel, Michael Gelbart, and Ryan Adams. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pp. 1746–1754. PMLR, 2014.

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, 2025. URL https://arxiv.org/abs/2506.01844.

Xudong Tan, Yaoxin Yang, Peng Ye, Jialin Zheng, Bizhe Bai, Xinyi Wang, Jia Hao, and Tao Chen. Think twice, act once: Token-aware compression and action reuse for efficient inference in vision-language-action models, 2025. URL https://arxiv.org/abs/2505.21200.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, Steven Bohez, Konstantinos Bousmalis, Anthony Brohan, Thomas Buschmann,

Arunkumar Byravan, Serkan Cabi, Ken Caluwaerts, Federico Casarini, Oscar Chang, Jose Enrique Chen, Xi Chen, Hao-Tien Lewis Chiang, Krzysztof Choromanski, David D'Ambrosio, Sudeep Dasari, Todor Davchev, Coline Devin, Norman Di Palo, Tianli Ding, Adil Dostmohamed, Danny Driess, Yilun Du, Debidatta Dwibedi, Michael Elabd, Claudio Fantacci, Cody Fong, Erik Frey, Chuyuan Fu, Marissa Giustina, Keerthana Gopalakrishnan, Laura Graesser, Leonard Hasenclever, Nicolas Heess, Brandon Hernaez, Alexander Herzog, R. Alex Hofer, Jan Humplik, Atil Iscen, Mithun George Jacob, Deepali Jain, Ryan Julian, Dmitry Kalashnikov, M. Emre Karagozler, Stefani Karp, Chase Kew, Jerad Kirkland, Sean Kirmani, Yuheng Kuang, Thomas Lampe, Antoine Laurens, Isabel Leal, Alex X. Lee, Tsang-Wei Edward Lee, Jacky Liang, Yixin Lin, Sharath Maddineni, Anirudha Majumdar, Assaf Hurwitz Michaely, Robert Moreno, Michael Neunert, Francesco Nori, Carolina Parada, Emilio Parisotto, Peter Pastor, Acorn Pooley, Kanishka Rao, Krista Reymann, Dorsa Sadigh, Stefano Saliceti, Pannag Sanketi, Pierre Sermanet, Dhruv Shah, Mohit Sharma, Kathryn Shea, Charles Shu, Vikas Sindhwani, Sumeet Singh, Radu Soricut, Jost Tobias Springenberg, Rachel Sterneck, Razvan Surdulescu, Jie Tan, Jonathan Tompson, Vincent Vanhoucke, Jake Varley, Grace Vesom, Giulia Vezzani, Oriol Vinyals, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Fei Xia, Ted Xiao, Annie Xie, Jinyu Xie, Peng Xu, Sichun Xu, Ying Xu, Zhuo Xu, Yuxiang Yang, Rui Yao, Sergey Yaroshenko, Wenhao Yu, Wentao Yuan, Jingwei Zhang, Tingnan Zhang, Allan Zhou, and Yuxiang Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning, 2026a. URL https://arxiv.org/abs/2509.05614.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, Xuhong Huang, Pei Lin, Junyang Lin, Dayiheng Liu, Shuai Bai, Jingren Zhou, Jiazhao Zhang, Haoqi Yuan, Gengze Zhou, Hang Yin, Ye Wang, Yiyang Huang, Zixing Lei, Wujian Peng, Delin Chen, Yingming Zheng, Jingyang Fan, Xianwei Zhuang, Xin Zhou, Haoyang Li, Anzhe Chen, Tong Zhang, Xuejing Liu, Yuchong Sun, Ruizhe Chen, Zhaohai Li, Chenxu Lü, Zhibo Yang, Tao Yu, and Xionghui Chen. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments, 2026b. URL https://arxiv.org/abs/2605.30280.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model, 2026. URL https://arxiv.org/abs/2601.18692.

Piotr Wyrwiński, Kacper Dobek, and Krzysztof Krawiec. Structure over pixels: Learning variable-length visual programs, 2026. URL https://arxiv.org/abs/2605.27696.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching, 2025. URL https://arxiv.org/abs/2502.02175.

Wilson Yan, Volodymyr Mnih, Aleksandra Faust, Matei Zaharia, Pieter Abbeel, and Hao Liu. Elastictok: Adaptive tokenization for image and video, 2025. URL https://arxiv.org/abs/2410.08368.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *Advances in Neural Information Processing Systems*, 38:40891–40914, 2026.

Yifan Ye, Jiaqi Ma, Jun Cen, and Zhihe Lu. Token expand-merge: Training-free token compression for vision-language-action models. *IEEE Robotics and Automation Letters*, 2026.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. *Advances in Neural Information Processing Systems*, 37:128940–128966, 2024.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *International Conference on Machine Learning*, 2025.

Ligeng Zhu. THOP: PyTorch-OpCounter. GitHub repository, 2022. URL `https://github.com/Lyken17/pytorch-OpCounter`.

A Pretraining Dataset Curation

Source datasets and camera selection. We construct the Stage 1 pretraining corpus from the 27 Open X-Embodiment datasets included in OpenVLA's `oxe_magic_soup_plus` collection. We read the training split of each dataset in RLDS format using pinned dataset versions. For each episode, we extract a single RGB stream specified by the dataset's primary-camera configuration. This is generally an external view, although some datasets provide an alternative viewpoint; for example, DOBBE uses a wrist-camera stream. We use the collection's dataset membership without applying OpenVLA's mixture-sampling weights. Figure 4 shows the number of cached frames contributed by each dataset.

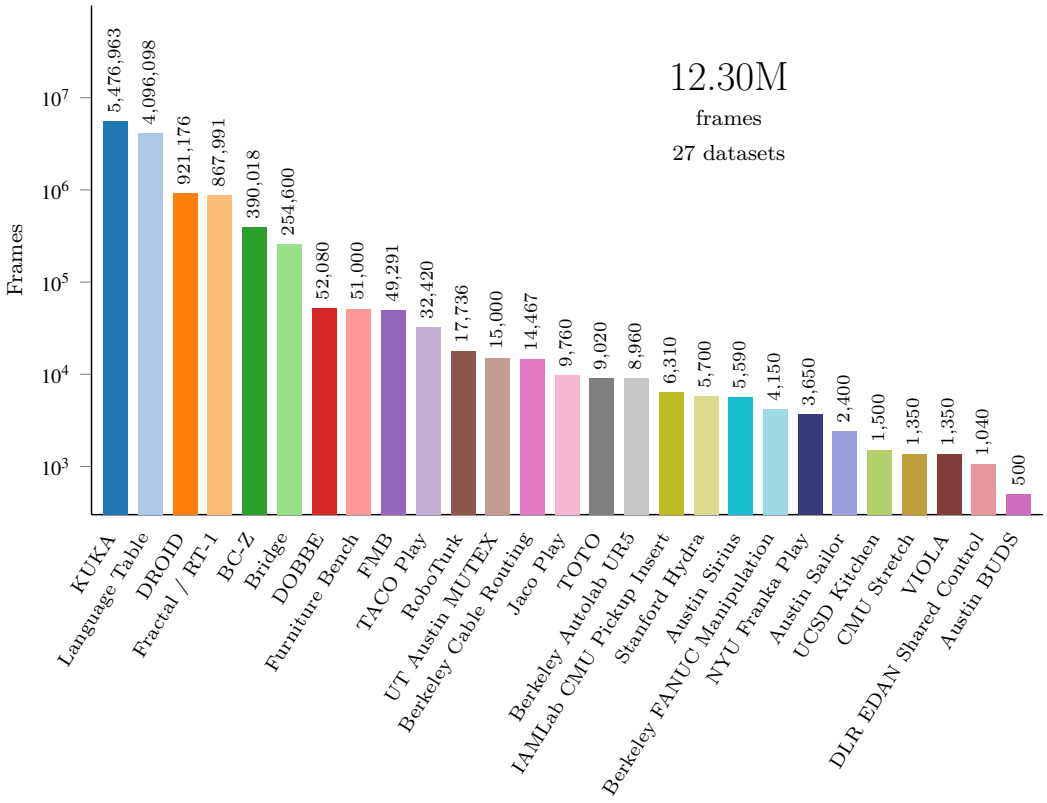


Figure 4: **Stage 1 pretraining dataset composition.** The 27 datasets contribute a total of 12,300,120 cached frames before the validation split. Datasets are ordered by frame count, with exact counts above the bars and a logarithmic vertical axis.

Temporal sampling and image preprocessing. We sample up to ten frames per episode in two passes. The first pass selects five approximately evenly spaced timesteps, including the first and last observations; episodes containing fewer than five timesteps contribute all available frames. The second pass preserves the initial selections and adds up to five distinct timesteps within the intervening gaps, prioritizing wider gaps and distributing additional samples within each gap. Short episodes and duplicate indices introduced by rounding can yield fewer than ten frames.

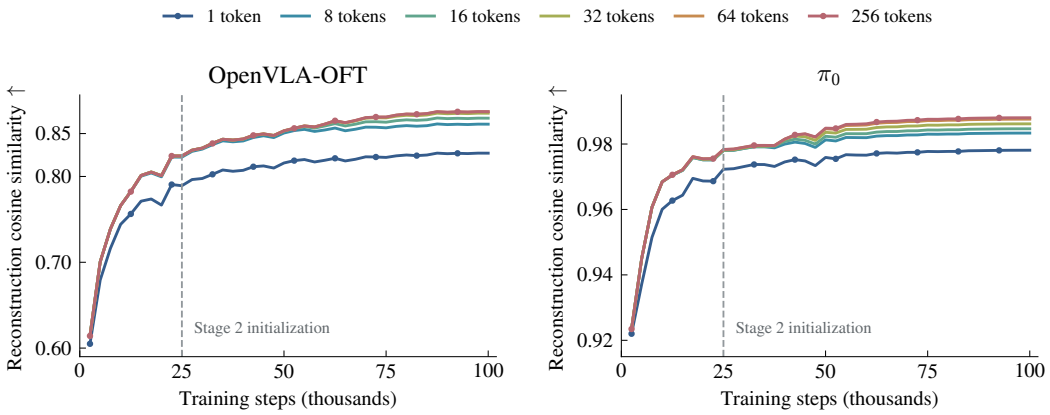


Figure 5: **Stage 1 reconstruction curves.** Held-out reconstruction cosine similarity for OpenVLA-OFT (left) and π_0 (right), evaluated every 2,500 steps at six token budgets per camera view. Dashed lines mark the 25,000-step checkpoints used for downstream initialization.

We convert each selected image to RGB, resize it directly to 224×224 using bicubic interpolation, and store it as a JPEG with quality 90. Each image receives a key containing its dataset, episode identifier, and timestep, preserving its provenance across extraction and feature caching. We exclude empty episodes and images that cannot be decoded or processed. For truncated shards, we retain only frames successfully read and encoded before the failure. We do not additionally filter episodes by task success or apply image-content deduplication.

Backbone-specific feature extraction. We construct separate feature caches for the two VLA backbones using their pretrained visual encoders. For OpenVLA-OFT, we extract the fused DINOv2 and SigLIP patch features from OpenVLA-7B, yielding a 256×2176 feature sequence per image. For π_0, we extract the SigLIP output before the multimodal projector, yielding a 256×1152 sequence. Images undergo the corresponding encoder-specific preprocessing before feature extraction. Both caches store features in bfloat16 together with their image keys. Stage 1 training reads these cached features directly, keeping the visual encoders fixed and applying no additional image augmentation.

B Stage 1 Reconstruction Curves

Figure 5 shows held-out feature-reconstruction cosine similarity during Stage 1 pretraining for OpenVLA-OFT and π_0. Reconstruction improves overall across the evaluated token budgets, with larger budgets achieving higher similarity later in training. For both backbones, we use the 25,000-step tokenizer checkpoint for LIBERO, LIBERO-Plus, and SIMPLER, as we empirically found that it provides better initialization for downstream policy training than later checkpoints. The full 100,000-step trajectories show continued improvements in reconstruction beyond this initialization point.

C Detailed Results Across Token Budgets

Tables 5, 6, and 7 report success rates for Flexi-Act with OpenVLA-OFT and π_0 on LIBERO, LIBERO-Plus, and SIMPLER at token budgets from 1 to 128 per camera view, alongside the uncompressed baselines. Performance largely saturates beyond 32 tokens per view, as this budget already recovers most of the uncompressed baseline's performance.

Table 5: LIBERO success rates (%) across token budgets.

Tokens/view	Spatial	Object	Goal	Long	Avg.
OpenVLA-OFT backbone					
Uncompressed	97.6	98.4	97.9	94.5	97.1
1	97.4	98.5	97.5	93.1	96.6
4	97.6	98.4	97.5	93.6	96.8
8	97.5	98.8	97.9	94.0	97.1
16	98.0	99.0	98.2	94.2	97.4
32	97.9	99.1	98.2	94.3	97.4
64	98.1	99.0	98.4	94.2	97.4
128	98.1	99.2	98.5	94.3	97.5
π_0 *backbone*					
Uncompressed	96.8	98.8	95.8	85.2	94.2
1	96.6	98.3	95.2	83.9	93.5
4	96.6	98.2	95.4	84.0	93.6
8	96.9	98.6	95.7	84.6	94.0
16	96.9	98.7	95.8	85.0	94.1
32	97.1	98.7	95.7	85.4	94.2
64	97.0	98.9	96.0	85.5	94.4
128	97.3	99.0	95.9	85.5	94.4

Table 6: LIBERO-Plus success rates (%) across token budgets and perturbation categories. Total denotes the task-count-weighted overall success rate.

Tokens/view	Camera	Robot	Language	Light	Background	Noise	Layout	Total
OpenVLA-OFT backbone								
Uncompressed	92.8	30.3	85.8	94.9	93.9	89.3	77.6	79.6
1	86.1	38.6	69.7	95.5	95.9	93.4	78.4	78.4
4	86.3	38.8	69.8	95.5	96.0	93.6	78.3	78.5
8	86.4	39.0	70.3	95.8	96.1	93.5	78.5	78.7
16	86.9	39.3	70.5	96.2	96.1	93.7	78.6	78.9
32	87.0	39.3	70.6	96.1	96.3	93.6	78.8	79.0
64	86.9	39.4	70.8	96.4	96.3	93.9	78.8	79.1
128	87.2	39.5	70.8	96.7	96.8	94.1	79.1	79.3
π_0 *backbone*								
Uncompressed	54.5	18.2	79.1	90.2	85.5	89.9	77.6	69.2
1	48.1	21.3	74.1	88.8	84.1	92.2	74.1	67.4
4	48.5	21.9	75.2	88.6	84.2	92.8	74.9	68.0
8	49.4	22.5	75.9	89.7	85.0	93.3	75.8	68.7
16	50.1	23.3	76.1	90.1	85.1	93.9	76.3	69.2
32	50.3	23.2	76.3	90.2	85.5	93.8	76.4	69.3
64	50.2	23.5	76.4	90.0	85.6	93.9	76.3	69.4
128	50.5	23.7	76.4	90.5	85.5	93.9	76.5	69.5

Table 7: SIMPLER success rates (%) across token budgets on the Google Robot tasks under the Visual Matching protocol.

Tokens/view	Pick Coke Can	Move Near	Open/Close Drawer	Avg.
OpenVLA-OFT backbone				
Uncompressed	69.7	68.3	53.2	64.6
1	61.3	63.4	51.1	59.0
4	63.6	65.1	52.0	60.7
8	66.2	66.5	52.7	62.6
16	69.6	68.0	53.3	64.4
32	69.7	68.1	53.4	64.5
64	69.7	68.1	53.3	64.5
128	69.8	67.9	53.4	64.5
π_0 *backbone*				
Uncompressed	62.2	59.0	44.1	56.0
1	50.9	53.7	41.1	49.0
4	54.3	55.4	42.1	51.2
8	58.1	58.1	42.9	53.8
16	61.9	58.9	44.3	55.9
32	62.1	59.0	44.3	56.0
64	62.0	59.1	44.5	56.1
128	62.4	59.1	44.5	56.2

D Inference Latency

We measure full-model and LLM-policy inference latency for OpenVLA-OFT across token budgets (Table 8). Flexi-Act reduces full-model latency from 159.9 ms to 75.2–78.0 ms, providing a 2.05–2.13× speedup. The separately measured LLM-policy latency decreases from 138.4 ms to 33.4–36.2 ms, corresponding to a 3.83–4.14× speedup. Increasing the budget from 1 to 16 tokens per camera view adds only 2.8 ms to full-model latency.

Table 8: Median GPU forward latency on an NVIDIA RTX A5000 with batch size 1 and two camera images, measured over 100 iterations after 20 warmup iterations. Parentheses indicate speedups relative to the uncompressed OpenVLA-OFT baseline.

Model	Tokens/view	Full model (ms) ↓	LLM policy (ms) ↓
OpenVLA-OFT	256	159.9 (1.00×)	138.4 (1.00×)
Flexi-Act	1	75.2 (2.13×)	33.4 (4.14×)
	4	75.9 (2.11×)	34.3 (4.04×)
	8	76.6 (2.09×)	34.9 (3.96×)
	16	78.0 (2.05×)	36.2 (3.83×)

E Architecture and Training Hyperparameters

Table 9 summarizes the tokenizer architecture and Stage 1 pretraining, and Table 10 gives the Stage 2 settings for LIBERO, LIBERO-Plus, and SIMPLER. All Stage 1 and Stage 2 training runs use eight NVIDIA H200 GPUs. Stage 1 uses the cached features from Appendix A and the cosine reconstruction objective from Section 4.2; Stage 2 jointly trains the tokenizer and policy, with the reconstruction decoder discarded. In both stages, we sample a token budget uniformly from $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$ per example; Stage 2 shares this budget across camera views. All budgets count tokens per view, and all budgets for a given policy are evaluated using the same checkpoint, retaining only the selected prefix at inference. We provide simplified Python implemen-

tations of Stage 1 tokenizer pretraining and Stage 2 policy fine-tuning for OpenVLA-OFT in the supplementary material.

Table 9: Stage 1 architecture and pretraining settings for both backbones. Stage 2 uses the 25,000-step checkpoint of the full 100,000-step learning-rate schedule.

Setting	OpenVLA-OFT	π_0
Tokenizer and training-only reconstruction decoder		
Visual feature / reconstruction shape	256×2176	256×1152
Tokenizer input projection	$2176 \rightarrow 1152$	None
Tokenizer layers / width / attention heads	18 / 1152 / 18	18 / 1152 / 18
Tokenizer attention	Bidirectional	Bidirectional
Feed-forward activation / expansion	SwiGLU / 4	SwiGLU / 4
Transformer normalization / QK normalization	RMSNorm / disabled	RMSNorm / disabled
Learned registers / bottleneck width	256 / 1152	256 / 1152
Bottleneck type / patch positions	Continuous / learned	Continuous / learned
Reconstruction decoder layers / width / attention heads	4 / 1024 / 16	4 / 1024 / 16
Reconstruction decoder attention	Self- and cross-attention	Self- and cross-attention
Reconstruction decoder feed-forward expansion	4	4
Reconstruction decoder patch queries	256 learned queries	256 learned queries
Reconstruction decoder query and key/value positions	Learned	Learned
Pretraining optimization		
Reconstruction objective	Cosine	Cosine
Optimizer	AdamW	AdamW
Adam (β_1, β_2) / ϵ	$(0.9, 0.99)$ / 10^{-8}	$(0.9, 0.99)$ / 10^{-8}
Weight decay	0.01	0.01
Tokenizer / reconstruction decoder peak learning rate	3×10^{-4} / 10^{-3}	3×10^{-4} / 10^{-3}
Global batch size	2048	2048
Learning-rate schedule	Cosine to zero	Cosine to zero
Warmup / schedule length (steps)	$1{,}000$ / $100{,}000$	$1{,}000$ / $100{,}000$
Checkpoint used for Stage 2 (steps)	25,000	25,000
Gradient-norm clipping / precision	1.0 / bfloat16	1.0 / bfloat16
Visual encoder / additional augmentation	Frozen / none	Frozen / none
Training seed	0	0
Checkpoint interval (steps)	2,500	5,000

Initialization. Both OpenVLA-OFT and π_0 use their respective reconditioned Stage 1 tokenizers at step 25,000 for LIBERO, LIBERO-Plus, and SIMPLER. These tokenizers use fixed input RMS normalization, Xavier initialization for the input projection (when present) and tokenizer transformer linear layers, and mask-embedding initialization standard deviation 0.02.

Table 10: Stage 2 settings across all three benchmarks. Settings describe Flexi-Act; uncompressed π_0 baselines share the policy optimization and durations, with the tokenizer omitted.

Setting	OpenVLA-OFT	π_0
Shared settings		
Base policy checkpoint	OpenVLA-7B	π_0 base
Action objective	L1 regression	Flow matching
Policy adaptation	LoRA + full heads[a]	Full fine-tuning
Tokenizer adaptation	Full fine-tuning	Full fine-tuning
LoRA rank / α / dropout	32 / 16 / 0	—
Optimizer	AdamW	AdamW
Adam (β_1, β_2) / ϵ	$(0.9, 0.95)$ / 10^{-8}	$(0.9, 0.95)$ / 10^{-8}
Weight decay / global batch size	0.01 / 64	10^{-10} / 32
Tokenizer learning rate / policy learning rate	0.1	0.1
Gradient-norm clipping / precision	1.0 / bfloat16	1.0 / bfloat16
Image augmentation	Enabled	Enabled
Checkpoint interval (steps)	10,000	5,000
LIBERO: separate policy per suite		
Training steps	150,000	30,000
Policy / tokenizer peak learning rate	5×10^{-4} / 5×10^{-5}	2.5×10^{-5} / 2.5×10^{-6}
Linear warmup steps	2,500	1,000
Learning-rate schedule	$\times 0.1$ at 100k	Cosine to $0.1\times$ peak
Final policy learning rate	5×10^{-5}	2.5×10^{-6}
Training seed	69	42
LIBERO-Plus: continuation from mixed LIBERO[b]		
Mixed-LIBERO initialization step	150,000	30,000
Additional training steps	150,000	50,000
Policy / tokenizer peak learning rate	5×10^{-4} / 5×10^{-5}	5×10^{-4} / 5×10^{-5}
Linear warmup steps	2,500	0
Learning-rate schedule	Cosine to zero	Constant
Optimizer state	Reset	Reset
Training seed	69	42
SIMPLER: fine-tuning on Fractal[c]		
Training steps	150,000	50,000
Policy / tokenizer peak learning rate	5×10^{-4} / 5×10^{-5}	5×10^{-4} / 5×10^{-5}
Linear warmup steps	2,500	0
Learning-rate schedule	$\times 0.1$ at 100k	Constant
Camera views / proprioceptive input	1 / omitted	1 / zero state

[a]LoRA adapts the visual encoder and language model; the visual projector, action head, and proprioceptive projector (when used) are fully trained. New policy components are initialized for base-checkpoint fine-tuning. [b]Mixed-LIBERO initialization uses the LIBERO settings above. OpenVLA-OFT merges the learned LoRA weights, retains the trained tokenizer and heads, and introduces fresh adapters. Both backbones train on mixed LIBERO-Plus demonstrations and use one checkpoint across perturbation dimensions. [c]SIMPLER evaluates Google Robot tasks; π_0 masks unused camera slots. The uncompressed OpenVLA-OFT Fractal baseline shares the listed policy settings and duration except for Adam $\beta_2 = 0.999$, no warmup, and no gradient clipping; it has no tokenizer. OpenVLA-OFT LIBERO reference results in Table 1 are from Compressor-VLA.