# FLOWER: Democratizing Generalist Robot Policies with Efficient Vision-Language-Action Flow Policies

**Moritz Reuss**[1] **Hongyi Zhou**[1] **Marcel Rühle**[1] **Ömer Erdinç Yağmurlu**[1]
**Fabian Otto**[2] **Rudolf Lioutikov**[1]

[1]Intuitive Robots Lab, Karlsruhe Institute of Technology, Germany [2]Microsoft Research

**Abstract:** Developing efficient Vision-Language-Action (VLA) policies is crucial for practical robotics deployment, yet current approaches face prohibitive computational costs and resource requirements. Existing diffusion-based VLA policies require multi-billion-parameter models and massive datasets to achieve strong performance. We tackle this efficiency challenge with two contributions: intermediate-modality fusion, which reallocates capacity to the diffusion head by pruning up to $50\%$ of LLM layers, and action-specific Global-AdaLN conditioning, which cuts parameters by $20\%$ through modular adaptation. We integrate these advances into a novel 950 M-parameter VLA called FLOWER. Pretrained in just 200 H100 GPU hours, FLOWER delivers competitive performance with bigger VLAs across 190 tasks spanning ten simulation and real-world benchmarks and demonstrates robustness across diverse robotic embodiments. In addition, FLOWER achieves a new SoTA of 4.53 on the CALVIN ABC benchmark. Demos, code and pretrained weights are available at `https://intuitive-robots.github.io/flower_vla/`.

**Keywords:** Imitation Learning, VLA,Language-conditioned Manipulation

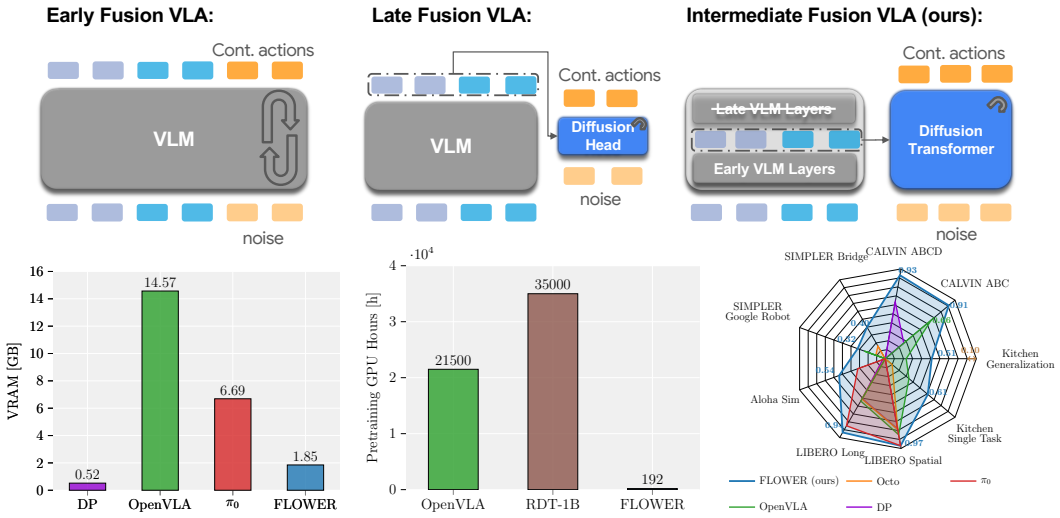
## 1 Introduction

Generalist robotic manipulation policies that execute diverse tasks across different embodiments remain a key goal in robotics. Recent advances in Imitation Learning (IL) have made significant progress toward this vision, particularly along the direction of generalist Vision-Language-Action-Model (VLA) Policies [2, 1, 3]. VLAs fine-tune pretrained Vision-Language-Models (VLMs) to generate robot actions from free-form language commands [2, 1, 3]. These models commonly adopt discrete [4, 1, 5, 6] or diffusion-based objectives [7, 8, 9] for action prediction. In particular, diffusion- and flow-based action generation excel at modeling complex, multimodal action distributions and have been successfully adopted for VLAs [8, 7, 10]. While VLAs offer many advantages, they have a major limitation: Current VLAs such as OpenVLA [1] and RDT-1B [8] contain several billion parameters and they require a lot of compute for pretraining, fine-tuning and real-robot deployment. This barrier limits access to more diverse research in the field.

In this work, we present contributions and insights that result in an efficient VLA policy that contains fewer than 1 billion parameters while matching current SoTA VLAs on 190 tasks across 10 benchmarks with 4 embodiments. As illustrated in Figure 1, dedicating most parameters to a deep, off-the-shelf VLM backbone forces the diffusion head to be severely under-parameterized for modeling rich, multimodal robot trajectories; conversely, shrinking the VLM to free up capacity strips away semantic features essential for robust instruction conditioning. Moreover, relying on the full VLM for denoising significantly slows convergence during training and increases inference latency.

Correspondence to: moritz.reuss@kit.edu

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.




(a) **Small GPU Memory** (b) **Less Compute** (c) **Strong Performance**

Figure 1: **Intermediate fusion for efficient VLA policies.** Our fusion strategy (top-right) strategically prunes VLM layers while enhancing Flow Transformer capacity in parameter-constrained settings. This approach informs FLOWER, a novel, $950M$ VLA that achieves competitive performance across 10 benchmarks using only $1\%$ of the pretraining compute of models like OpenVLA [1], while maintaining a small memory footprint across diverse embodiments and action spaces (bottom).

To resolve this budget–tradeoff, we introduce four key contributions. First, we propose *intermediate-level fusion*: we prune between $30\%$ and $50\%$ of the pretrained VLM's layers and condition our Flow Transformer on the resulting intermediate embeddings, thereby retaining semantic grounding while reclaiming parameters. Second, we develop a *global action-space AdaLN* mechanism: an action-specific LayerNorm controller within the diffusion transformer that reduces head parameters by $20\%$ without any loss of accuracy or expressivity. Third, we provide extensive ablations across multiple benchmarks that evaluate design choices and yield additional insights into which VLM architectures and pretraining objectives are best suited for efficient VLA adoption.

Leveraging these contributions and insights, we present our fourth contribution: **Florence With Embodied Flow (FLOWER)**, a 950M parameter VLA policy that converges quickly, cuts pretraining cost on heterogeneous robotics data by $99\%$, and yields competitive performance with current VLAs across 190 tasks in 10 benchmarks across simulation and real world settings.

## 2 Related Work

Early imitation-learning studies demonstrated that policy performance scales with dataset size. For example, Pinto and Gupta [11] used over 50K grasps and observed steady improvements in manipulation proficiency. Building on this, the OXE benchmark provided 1.4M trajectories across more than 20 robot embodiments, enabling research into generalist policies [6]. Several diffusion-policy approaches directly train on OXE without leveraging pretrained VLMs. Octo [3] applies a transformer-based diffusion policy to delta end-effector actions but lacks a pretrained vision–language encoder and model capacity, which limits its generalization as observed on benchmarks such as SIMPLER [12]. RDT introduces a 1.2B-parameter diffusion transformer paired with an 11.4B-parameter pretrained language encoder model and Vision Transformer (ViT). However, its one-month pretraining on 48 A100 GPUs highlights the need for more computationally efficient VLA pretraining [8]. Addressing this need, our contributions yield versatile VLAs in only 200 GPU hours of pretraining.

To address generalization, recent methods incorporate large pretrained VLMs into the policy. OpenVLA fine-tunes a 7.7B-parameter VLM for discrete end-effector action prediction, yet its size makes real-robot deployment challenging [1]. Several other VLAs apply discrete action prediction [2, 13, 5, 14, 15, 16]. RoboDual improves upon OpenVLA, by integrating a small diffusion transformer with OpenVLA via asynchronous updates, while Latent Bridge uses a fine-tuned VLM to generate latent commands that condition a diffusion policy [17, 18]. Similar to FLOWER, $\pi_0$ [7] and GR00T-N1 [9], also introduce generalist flow-based VLA with more than 2B parameters, which are both trained on a closed-source cross-embodiment datasets. Despite their effectiveness, these approaches retain very large VLMs with high-memory requirements and slow convergence. In contrast, FLOWER matches these methods with less than 1 billion parameters and significantly lower memory and fully open source training.

Efforts to reduce model size and finetuning overhead include using smaller VLM backbones or alternative fusion strategies. TinyVLA attaches a compact diffusion head to a lightweight VLM with a late-fusion approach and no pretraining [19]. Fusion strategies range from early fusion, that merges raw VLM outputs with the action generator at the input stage, to late fusion, where modalities are processed separately and combined only after independent streams [19, 8, 20, 21]. In contrast, intermediate fusion (our contribution) injects mid-level VLM tokens into the Flow Transformer, allowing selective pruning of the VLM while preserving semantic richness. While DeerVLA [22] has explored early exit strategies for continuous action prediction VLAs, FLOWER significantly surpasses it across benchmarks by wide margins thanks to our novel flow-based fusion design. Beyond model architecture, the choice of data sources and automated data collection are key to pretraining robust policies. Concurrent work has also explored intermediate-fusion [9] without any ablation and insights on the impact. FLOWER advances efficient VLAs using intermediate-modality fusion and global-adaln conditioning, enabling faster training and inference while achieving SoTA performance on diverse benchmarks.

## 3 Method

We learn an efficient, generalist policy $\pi_\theta$ that generates actions conditioned on state $s_t$, text goal tokens $g_t$, and meta-embodiment information $e_i$. This task is challenging due to heterogeneity in action spaces (varying degrees-of-freedom, control modes), observation spaces (different sensors), and task specifications (diverse language goals). Given trajectories from multiple embodiments, we learn a unified policy that generalizes across robots, tasks, and observation formats by maximizing $\mathcal{L}_{\text{IL}} = \mathbb{E}_{(s,a,g,e)\sim\mathcal{D}}[\log \pi_\theta(a \mid s, g, e)]$.

### 3.1 Intermediate Modality Fusion Vision-Language-Action-Models

On a high level, flow-based VLA consists of two main components: A VLM pretrained on internet-scale vision and text data to encode state information and text and a flow prediction module to generate a sequence of actions conditioned on the current context from Gaussian noise.

We propose to fuse pretrained vision–language representations with Flow Transformer embeddings at an intermediate stage to balance semantic depth and computational efficiency. Prior work in Large-Language-Model (LLM) explainability shows that features from the penultimate quarter of transformer layers capture broad semantics, whereas final layers specialize in next-token prediction [23]. Accordingly, this motivates our VLA fusion strategy to extract hidden states from an intermediate layer of the VLM backbone after jointly encoding visual (ViT) and textual tokens. These intermediate features preserve rich context without overspecialization for next-token prediction and reduced computational cost to enable more efficient VLA design. We prune the VLM based on its architecture: For **Encoder–Decoder VLMs** (Florence-2 [24]), we remove the full decoder, keeping only the encoder LLM layers. This reduces the number of layers by $50\%$ while increasing its performance and efficiency. For **Decoder-Only VLMs** (SmolFlow2-Video [25]), we drop the final $30\%$ of transformer layers. This targeted pruning cuts 20–35% of parameters and reduces per-step

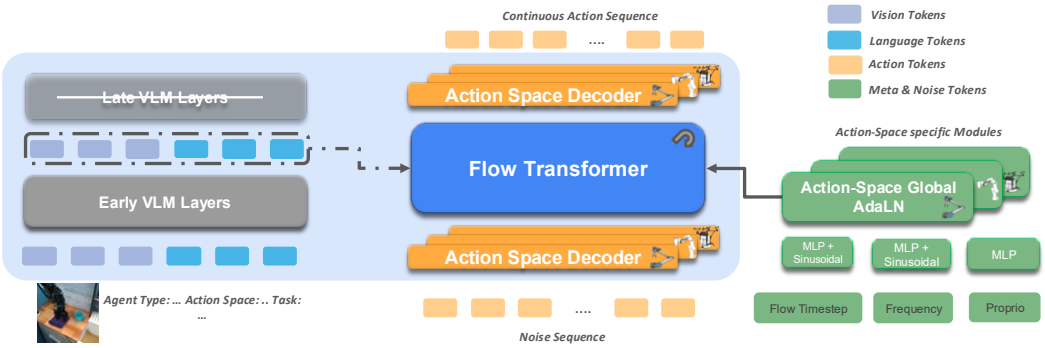



Figure 2: **FLOWER architecture.** A fine-tuned VLM processes multimodal inputs and integrates intermediate features into a Flow Transformer via cross-attention. The model predicts velocity fields using action-space Global AdaLN-Zero conditioning with embodiment and temporal metadata.

latency. In Subsection 4.1, we provide ablations to verify our intermediate fusion design. The full architecture is visualized in Figure 2.

Next, we take the projected VLM latent tokens and inject them into the Flow Transformer via cross-attention. Specifically, we map the VLM hidden states through a linear layer followed by RMSNorm [26] for increased stability. This design conditions each Flow layer on semantically rich VLM features, preserving spatial and contextual structure and enabling faster policy convergence without the late fusion overhead.

### 3.2 Cross-Action Space Flow Transformer

We design a novel, efficient Flow Transformer to handle heterogeneous action spaces efficiently. In particular, we introduce *Action-Space Global-AdaLN-Zero*, a unified normalization layer that conditions each transformer block on both temporal signals (e.g., flow time step) and per-action-type embeddings without incurring large parameter overheads, as illustrated in Figure 3.

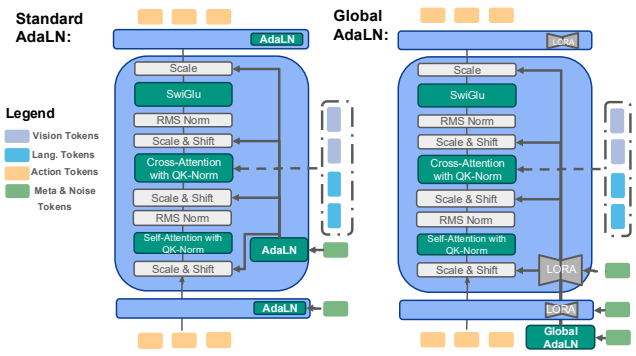



Figure 3: Comparison of standard DiT blocks and our proposed Global AdaLN with layer-specific Lora adapters.

Standard AdaLN-Zero uses distinct scale-and-shift parameters per layer, adding up to 30% extra parameters in diffusion transformers [27]. In contrast, *Action-Space Global-AdaLN-Zero* shares a single set of modulation weights across all layers, while generating unique modulation signals for each action category (e.g., delta-EEF vs. joint angle), initialized with zeros for stable training. This reduces parameter count by over 20% compared to naive AdaLN-Zero, yet preserves adaptation to action-space statistics. Given this lower per-layer expressiveness, we additionally inject lightweight LoRA adapters into each layer. This provides fine-grained, layer-specific modulation with only a few extra parameters per block. Additionally, each action type uses a small encoder/decoder to map actions into/out of the transformer latent space, enabling consistent handling of differing action dimensions without sacrificing weight sharing.

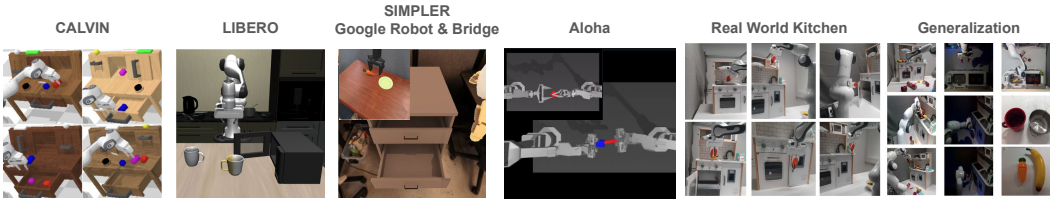



Figure 4: **Simulation Environments used to test FLOWER.** From left to right: *CALVIN* [34], *LIBERO* [35], *SIMPLER* [12] with the *Bridge* and *Google Robot* variants and *Aloha Simulation Benchmark* [36]. Real world multi-task kitchen setup and generalization experiments with cluttered scenes, different lightning and novel objects.

### 3.3 Rectified Flow for Action Generation

We utilize Rectified Flow for generating actions. Flow models use straight-line velocity fields between noise and data distributions [28, 29]. This approach reduces inference computation while maintaining expressiveness, which crucial for policies where latency matters. For conditional action distribution $\pi_\theta\left(\bar{\boldsymbol{a}}_{n,k}|\bar{\boldsymbol{s}}_n, \boldsymbol{g}, \boldsymbol{e}\right)$, the trajectory interpolation follows:

$$z_t = (1-t)\bar{\boldsymbol{a}}_{n,k} + tz_1, \quad z_1 \sim \mathcal{N}(0, I), \tag{1}$$

where $t \in [0,1]$ is normalized flow time sampled from $t \sim \sigma(\mathcal{N}(0,I))$, $\bar{\boldsymbol{a}}_{n,k} \in \mathbb{R}^{d_a}$ is the ground truth action sequence, and $z_1$ is standard Gaussian noise. The model optimizes:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,z_1}\left[\|z_1 - \bar{\boldsymbol{a}}_{n,k} - v_\theta(z_t, t, \bar{\boldsymbol{s}}_n, \boldsymbol{g}, \boldsymbol{e})\|^2\right], \tag{2}$$

where $v_\theta$ is the flow model conditioned on state $\bar{\boldsymbol{s}}$, language goal $\boldsymbol{g}$, and embodiment $\boldsymbol{e}$. Inference requires only $N = 4$ denoising steps for single-arm and $N = 8$ for high-frequency dual-arm settings.

### 3.4 FLOWER: Efficient Flow-based Vision-Language-Action Models

Leveraging our contributions and insights in intermediate fusion and Global-AdaLN, we present FLOWER: a compact, efficient VLA policy with exceptional performance-to-parameter ratio. FLOWER uses half of the Florence-2-L VLM as its primary backbone, which our ablations show provides optimal performance for robotic manipulation tasks. The model employs an 18-layer Flow Transformer with 1024 latent dimension. Together with the action-specific modules, FLOWER has 947M parameters in total and only requires 1.85 GB of VRAM. This architecture achieves the computational efficiency required for real-time deployment

**Cost-Effective Pretraining Setup.** To enable fast, cost-efficient pretraining, FLOWER uses a small, carefully chosen "OXE-soup" of eight public robotic datasets (approximately $250k$ trajectories total). We prioritize broad scene and embodiment diversity—including Franka Pandas and XARMs—and, inspired by Lin et al. [30]. $75\%$ of our data samples from Droid [31], Google Robot [13], and BridgeV2 [32]. Unlike most OXE datasets, these data are gathered in varied environments with rich distractions, backgrounds, and objects. We use an action chunk length of 20 and a single static image input [23, 33]. By training on a compact mixture of $74\%$ delta-EEF and $26\%$ single-arm joint-state data, we complete $360,000$ steps in 48 h ($\approx$ 200 GPU-hours). Extending the run yields no further gains (see App. A). To identify this optimum, we track zero-shot success on the Bridge SIMPLER benchmark [12], observing that FLOWER Bridge performance stagnates thereafter.

## 4 Evaluation

We aim to answer the following research questions in our experiments: **(RQ I)** How do the proposed design elements of our novel VLA architecture impact performance? **(RQ II)** Does our VLA-design deliver strong performance while significantly reducing computational demands compared to sota VLA policies across diverse settings? **(RQ III)** Can we train a VLA policy with less than 1B parameters that handles robot embodiments (e. g., single-arm vs. dual-arm, delta-EEF vs. joint angles)

| Fusion Strategy | Success Rate (%) | | | |
|---|---|---|---|---|
| | Florence-VLM | | SMol-VLM | |
| | C-ABC | L-Long | C-ABC | L-Long |
| Early | $57.1 \pm 5.3$ | $33.4 \pm 6.0$ | $25.8 \pm 3.9$ | $44.5 \pm 2.7$ |
| Inter. | $\mathbf{89.5 \pm 1.0}$ | $\mathbf{93.4 \pm 2.0}$ | $\mathbf{72.1 \pm 5.0}$ | $\mathbf{70.7 \pm 2.3}$ |
| Late | $71.2 \pm 2.2$ | $61.8 \pm 2.5$ | $66.3 \pm 2.0$ | $69.2 \pm 1.9$ |

Table 1: **Evaluation of different VLA fusion strategies.** Intermediate fusion yields the best performance across both VLM types.

| Layers | Model | |
|---|---|---|
| | C-ABC | L-Long |
| Full | $66.3 \pm 2.0$ | $69.2 \pm 1.9$ |
| 0.2 | $68.6 \pm 3.2$ | $71.8 \pm 3.7$ |
| 0.3 | $72.1 \pm 5.0$ | $70.7 \pm 2.3$ |
| 0.5 | $66.4 \pm 6.4$ | $62.5 \pm 3.5$ |

Table 2: **Comparing Layer Pruning** for CALVIN ABC and LIBERO Long.

and robustly transfers to unseen environments, novel objects, and varying conditions? We evaluate our method across more than 190 tasks in 10 different benchmarks to answer these questions.

### 4.1 Evaluation of Critical Design Decisions for Efficient Flow VLAs

In this section, we evaluate the key design decisions that impact the efficiency and performance of small Flow-based VLAs across two benchmarks: CALVIN ABC [34] and LIBERO-Long [35]. We train each policy variant with 3 seeds for up to $100k$ steps and report the average performance. While CALVIN-ABC focuses on generalization for free form instruction following to solve 34 different task, LIBERO-Long emphasizes long-horizon task completion. Both benchmarks are established for testing VLAs and provide useful insights into optimal design.

**Does intermediate fusion provide strong performance with higher efficiency?** The fusion strategy determines how the VLM's vision-language features integrate with the action prediction model. We compare three strategies: early fusion (combining noise token with VLM tokens at LLM level), our proposed intermediate fusion, and late fusion (using final VLM outputs to condition the flow prediction module). As shown in Table 1 and Table 2, intermediate fusion outperforms alternatives with both model variants and on both benchmarks. With Florence-VLM, it achieves $93.4\%$ success on LIBERO-Long, a 61 percentage point improvement over early fusion ($33.4\%$) and 21 points over late fusion ($73\%$). This confirms our hypothesis that intermediate-layer features provide an optimal balance of semantic richness and computational efficiency, addressing **(RQ I)**.

**What type of VLM backbone is best for efficient VLAs?** We explore different VLM backbones, comparing Florence-2-L [24] and SmolFlow-500M [25]. Beyond their architectural differences (Florence-2 being encoder-decoder and SmolFlow decoder-only), their pretraining objectives significantly differ. Florence-2 was trained on FLD-5B (5.4 billion annotations across 126 million images) with emphasis on object detection, segmentation and visual grounding, while SmolFlow's pretraining prioritizes general reasoning and language understanding capabilities across text, videos and images. This is representative for a standard VLM pretraining. Our experiments confirm that Florence-2's pretraining translates more effectively to robotic manipulation tasks, further addressing **(RQ I)**. We therefore use the Florence-2 variant of FLOWER for all subsequent experiments.

| **Variant** | **Avg. Len.** |
|---|---|
| FLOWER | **4.44±0.04** |
| + standard AdaLN | 4.43±0.03 |
| - Flow Head | 3.33±0.04 |
| - Custom LR | 4.40±0.05 |
| - No VLM train | 2.65±0.36 |
| + small Florence | 4.26±0.04 |
| - VLM | 3.42±0.07 |
| + Discrete Token | 1.12±0.12 |
| + Smaller Head | 2.60 ±0.09 |

Table 3: Average Sequence Lengths for FLOWER Ablations on CALVIN ABC.

**Does Global-AdaLN enable more efficient Flow Transformers?** Next, we compare our proposed Global-AdaLN against default AdaLN used in prior work [37, 38] for our Florence-based FLOWER model on CALVIN ABC. The results in Table 3 demonstrate that our proposed Global AdaLN enables relevant parameter reduction of $20\%$ without reducing the performance.

**Do we need a large-capacity Diffusion Transformer?** Next, we test FLOWER with a small action head using 384 latent dimensions and 6 layers. The final model only achieves significantly lower average performance of 2.60 compared to our design. This confirms that having a high capacity Diffusion Transformer is crucial for performance.

**What other design choices matter?** Finally, we test our VLA with several ablations variants using the Florence2 version on CALVIN ABC (Table 3). The smaller Florence reduces performance notably, while the frozen VLM has an even bigger negative impact on performance. In addition,

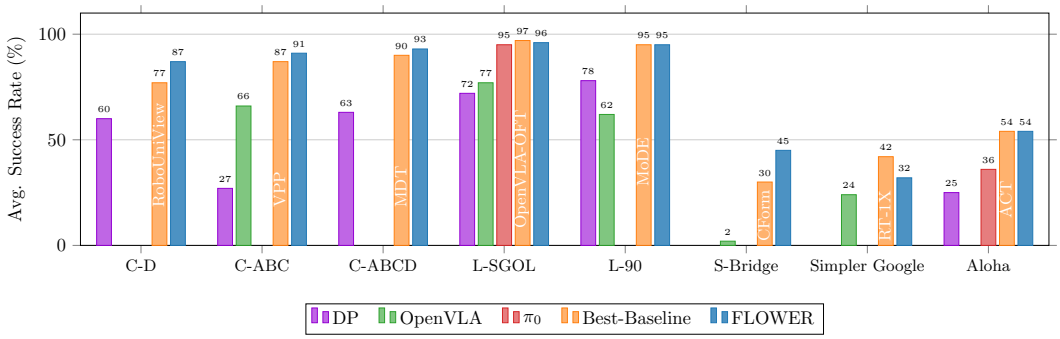



Figure 5: **Simulation Results for FLOWER** We report average results for various benchmarks against relevant baselines. For brevity we reduce the shown baselines to most relevant ones but provide detailed results for each benchmark (see Appendix B. C refers to CALVIN and L refers to LIBERO. SGOL refers to average results for LIBERO Object, Goal, Spatial and Long.

our custom Learning Rate scheduler increases performance. Finally, we compare FLOWER with an L1-prediction head like ACT [36] and a discrete token variant to demonstrate the importance of using flow prediction. These findings complete our analysis for **(RQ I)**.

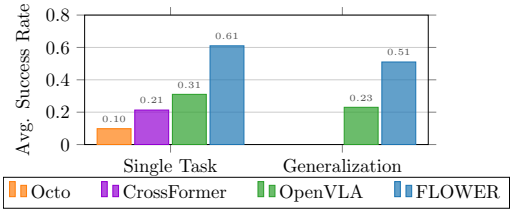
### 4.2 Simulation Experiments

Next, we take the best performing VLA variant, FLOWER, with the Florence-2-L backbone and pretrain it on our pretraining mix. After pretraining, we evaluate FLOWER across multiple simulation benchmarks to assess its performance, generalization capabilities, and adaptation to different robotic environments using finetuning.

**Benchmark Evaluation.** We evaluate FLOWER on four established benchmark suites representing diverse robotic manipulation challenges: **CALVIN** [34] features 34 tabletop manipulation tasks with a Franka Panda robot using delta end-effector control across four scene configurations (splits A-D). The dataset contains 24,000 language-annotated demonstrations. We evaluate three settings: CALVIN D, CALVIN ABC (zero-shot generalization), and CALVIN ABCD (scaling with more data). Performance is measured by success rates on sequential tasks and mean sequence length completion. All evaluations require policies to follow free-form language instructions and complete 5 tasks in sequence across 1,000 different instruction chains. **LIBERO** [35] tests a delta-EEF controlled Panda Robot across various scenes and challenges. We report results on four specialized variants with 10 tasks each (Long, Spatial, Object, and Goal) and separately on LIBERO-90, which requires policies to solve 90 different tasks in diverse scenes. Success is measured as the percentage of successful task completions across 50 trials per task (20 for LIBERO-90). **Aloha** [36] simulation tasks evaluate bi-manual joint state manipulation, requiring high-frequency control (50Hz), where tasks include cube transfer and peg insertion. **SIMPLER** [12] provides evaluation after pretraining on real2sim environments with Bridge and Google Robot variants.

**Baselines.** We compare our VLA against sota VLA policies and specialized approaches, using results reported in prior publications for fair comparison. Our primary comparisons are with OpenVLA [1] (7.7B parameters), $\pi_0$ [7] (3.3B parameters), and a standard Diffusion Policy using a CNN [39]. We provide detailed comparisons against additional relevant baselines for each benchmark in Appendix B, including video-based VLAs like VPP [20], and specialized policies like Baku [40].

**Results.** The results for these experiments are summarized in Figure 5. FLOWER consistently matches or surpasses all current SoTA approaches across all reported benchmarks. Notably, it surpasses OpenVLA on CALVIN and LIBERO by wide margins despite its smaller size and faster training. It also outperforms $\pi_0$ on the LIBERO and ALOHA benchmarks. Overall, these results demonstrate the versatility of FLOWER to adapt do diverse embodiments and task settings. These results confirm that FLOWER provides strong performance at low computational cost **(RQ II)** and demonstrates generalization to unseen environment settings **(RQ III)**.




(a) **Real World Results** for different Generalist Policies finetuned on a Franka Panda Kitchen Setup.

| Generalization Scenario | FLOWER | OpenVLA |
|---|---|---|
| Novel Object | **33.3%** | 10.0% |
| Flashlight | **50.0%** | 25.0% |
| BG Distractors | **69.5%** | 41.7% |
| New Tasks Composition | **51.1%** | 16.7% |
| Average | **51.0%** | 23.4% |

(b) **Generalization Results** comparing FLOWER and OpenVLA across different scenarios.

Figure 6: **Real World Results for Multi-Task and Generalization Experiments**

### 4.3 Real-World Evaluation and Generalization

Next, we want to verify our results from simulation in a real world setup. Therefore, we evaluate FLOWER in a real-world kitchen setting with a Franka Panda Robot across 20 distinct tasks involving various objects. The training data consists of 417 language-annotated trajectories from 45 minutes of human demonstrations collected via kinesthetic teaching. All Policies are finetuned in joint state space at 6 Hz. An overview of various tasks is shown in Figure 4 and we additionally provide rollouts in our project page. We compare FLOWER against several generalist VLA policies: Octo [3], OpenVLA [1], and CrossFormer [41], all finetuned on our dataset using recommended hyperparameters (see subsubsection B.2.1 for details).

**Results.** First, we test all policies on instruction following for all 20 different tasks. Each task is evaluated five times from randomized starting positions. As shown in Figure 6a, FLOWER achieves the highest average success rate (61%), doubling the performance of the second-best baseline, OpenVLA (31%). These results address **(RQ II)** by demonstrating FLOWER's strong performance compared to state-of-the-art baselines in real-world settings.

**Generalization Analysis.** We further test FLOWER's generalization capabilities against OpenVLA under challenging conditions: novel objects, flashlight-only lighting, background distractors, and new task compositions. As summarized in Figure 6b, FLOWER consistently outperforms OpenVLA across all scenarios, averaging 51.0% success compared to OpenVLA's 23.4%. These experiments verify that FLOWER can effectively adapt to unstructured, real-world variations **(RQ III)**, though it still faces challenges with fine manipulation in highly cluttered environments.

**Inference Efficiency.** We evaluated FLOWER against baselines on an RTX 4090 GPU (Table 4). FLOWER achieves a throughput of 311Hz, making it 8% faster than $\pi_0$ (288Hz), and 5007% faster than OpenVLA (6.1Hz). FLOWER has **the lowest memory footprint** among VLAs, using 27.6% of $\pi_0$'s memory and 12.7% of OpenVLA's, making it ideal for commodity hardware. These results strongly support **(RQ II)**.

| Method | Throughput (Hz)↑ | Latency (s)↓ | VRAM (MB)↓ |
|---|---|---|---|
| DP (0.26B) | 130.67 | 0.341 | **517** |
| OpenVLA (7B) | 6.09 | 0.164 | 14574 |
| $\pi_0$ (3.3B) | *288.11* | *0.104* | 6692 |
| **FLOWER+LF(1.15B)** | 287.36 | 0.055 | 2235 |
| **FLOWER (0.95B)** | **311.04** | **0.052** | *1848* |

Table 4: **Mean inference efficiency** (1000 steps in Bf16). All policies except OpenVLA use chunk length 50. *LF* refers to Late Fusion Ablation of FLOWER with the complete VLM.

## 5 Conclusion

We introduced several contributions for efficient VLAs: intermediate-level fusion that preserves semantic understanding while pruning 30-50% of VLM layers, and global action-space AdaLN that reduces transformer head parameters by 20% without compromising expressivity. These techniques enable compact yet powerful flow-based VLAs. This yields FLOWER, an efficient VLA that matches current state-of-the-art VLAs across 190 tasks in 10 benchmarks, despite requiring only 950M parameters and 200 GPU hours for pretraining.

## 6 Limitations

Despite its advantages, FLOWER still has several limitations. First, it relies on an iterative sampling procedure, which is inherently slower than a single forward pass from deterministic policies. Second, we have validated FLOWER primarily on three manipulation action spaces; its ability to generalize to other embodiments, such as mobile navigation or humanoid locomotion, remains unexplored and is an important direction for future work. Third, pretraining performance for zero-shot deployment on the SIMPLER Google Robot benchmark indicates that further improvements are needed. We hypothesize that the generalization tested in SIMPLER benefits from larger models. Fourth, although FLOWER is considerably smaller than most state-of-the-art VLA models, its $\approx 1$ B-parameter size may still present deployment challenges in low-resource or real-time settings. Fifth, eight out of our ten used benchmarks are conducted in simulation, limiting the extent to which our results can be taken as evidence of real-world generalization.

## 7 Acknowledgments

The work was funded by the German Research Foundation (DFG) – 448648559. The authors also acknowledge support by the state of Baden-Württemberg through HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the German Federal Ministry of Education and Research.

## References

[1] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022.

[3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. https://octo-models.github.io, 2023.

[4] X. Li, P. Li, M. Liu, D. Wang, J. Liu, B. Kang, X. Ma, T. Kong, H. Zhang, and H. Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024.

[5] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[6] O. X.-E. Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. https://arxiv.org/abs/2310.08864, 2023.

[7] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. $pi\_0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[8] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024.

[9] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[10] M. Reuss, J. Pari, P. Agrawal, and R. Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning, 2024.

[11] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[12] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[13] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[14] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

[15] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

[16] H. Etukuru, N. Naka, Z. Hu, S. Lee, J. Mehu, A. Edsinger, C. Paxton, S. Chintala, L. Pinto, and N. M. M. Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. *arXiv preprint arXiv:2409.05865*, 2024.

[17] Q. Bu, H. Li, L. Chen, J. Cai, J. Zeng, H. Cui, M. Yao, and Y. Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*, 2024.

[18] Y. Shentu, P. Wu, A. Rajeswaran, and P. Abbeel. From llms to actions: Latent codes as bridges in hierarchical robot control. *arXiv preprint arXiv:2405.04798*, 2024.

[19] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

[20] Y. Hu, Y. Guo, P. Wang, X. Chen, Y.-J. Wang, J. Zhang, K. Sreenath, C. Lu, and J. Chen. Video prediction policy: A generalist robot policy with predictive visual representations, 2024. URL https://arxiv.org/abs/2412.14803.

[21] J. Li, Y. Zhu, Z. Tang, J. Wen, M. Zhu, X. Liu, C. Li, R. Cheng, Y. Peng, and F. Feng. Improving vision-language-action models via chain-of-affordance. *arXiv preprint arXiv:2412.20451*, 2024.

[22] Y. Yue, Y. Wang, B. Kang, Y. Han, S. Wang, S. Song, J. Feng, and G. Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

[23] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, and J. Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[24] B. Xiao, H. Wu, W. Xu, X. Dai, H. Hu, Y. Lu, M. Zeng, C. Liu, and L. Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4818–4829, 2024.

[25] A. Marafioti, O. Zohar, M. Farré, M. Noyan, E. Bakouch, P. Cuenca, C. Zakka, L. B. Allal, A. Lozhkov, N. Tazi, V. Srivastav, J. Lochner, H. Larcher, M. Morlon, L. Tunstall, L. von Werra, and T. Wolf. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

[26] B. Zhang and R. Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[27] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

[28] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[29] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

[30] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.

[31] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[32] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[33] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[34] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

[35] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[36] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[37] M. Reuss, Ö. E. Yağmurlu, F. Wenzel, and R. Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *Robotics: Science and Systems*, 2024.

[38] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[39] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[40] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

[41] R. Doshi, H. R. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *8th Annual Conference on Robot Learning*, 2024.

[42] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

[43] N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[44] A. Henry, P. R. Dachapally, S. Pawar, and Y. Chen. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*, 2020.

[45] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

[46] X. Li, M. Liu, H. Zhang, C. Yu, J. Xu, H. Wu, C. Cheang, Y. Jing, W. Zhang, H. Liu, H. Li, and T. Kong. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[47] K. Black, M. Nakamoto, P. Atreya, H. Walke, C. Finn, A. Kumar, and S. Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[48] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

[49] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=gqCQxObVz2.

[50] Y. Tian, S. Yang, J. Zeng, P. Wang, D. Lin, H. Dong, and J. Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. *https://arxiv.org/abs/2412.15109*, 2024.

[51] F. Liu, F. Yan, L. Zheng, Y. Huang, C. Feng, and L. Ma. Robouniview: Visual-language model with unified view representation for robotic manipulation. *arXiv preprint 2406.18977*, 2024.

[52] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

[53] D. Driess, J. T. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Z. Ren, H. Walke, Q. Vuong, L. X. Shi, et al. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint arXiv:2505.23705*, 2025.

[54] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[55] R. Cadene, S. Alibert, A. Soare, Q. Gallouedec, A. Zouitine, and T. Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

[56] X. Jiang, P. Mattes, X. Jia, N. Schreiber, G. Neumann, and R. Lioutikov. A comprehensive user study on augmented reality-based data collection interfaces for robot learning. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 333–342, 2024.

[57] W. Huang, I. Mordatch, and D. Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

[58] T. Chen, A. Murali, and A. Gupta. Hardware conditioned policies for multi-robot transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[59] A. Patel and S. Song. Get-zero: Graph embodiment transformer for zero-shot embodiment generalization. *arXiv preprint arXiv:2407.15002*, 2024.

[60] J. H. Yang, D. Sadigh, and C. Finn. Polybot: Training one policy across robots while embracing variability. In *Conference on Robot Learning*, pages 2955–2974. PMLR, 2023.

[61] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[62] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

[63] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

[64] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[65] X. Jia, D. Blessing, X. Jiang, M. Reuss, A. Donat, R. Lioutikov, and G. Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. *arXiv preprint arXiv:2402.14606*, 2024.

[66] M. S. Albergo and E. Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

[67] N. Funk, J. Urain, J. Carvalho, V. Prasad, G. Chalvatzaki, and J. Peters. Actionflow: Equivariant, accurate, and efficient policies with spatially symmetric flow matching. *arXiv preprint arXiv:2409.04576*, 2024.

[68] F. Zhang and M. Gienger. Affordance-based robot manipulation with flow matching. *arXiv preprint arXiv:2409.01083*, 2024.

[69] M. Braun, N. Jaquier, L. Rozo, and T. Asfour. Riemannian flow matching policy for robot motion learning. *arXiv preprint arXiv:2403.10672*, 2024.

[70] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

| | SIMPLER | CALVIN | LIBERO | Aloha | Real World Kitchen |
|---|---|---|---|---|---|
| Action Space Encoders | | | 2-layered MLP | | |
| Action Space Decoders | | | Linear Attention | | |
| Number of Flow-T Layers | | | 18 | | |
| Latent Dimension | | | 1024 | | |
| Number of Heads | | | 16 | | |
| Position Embedding | | | 1D Rope | | |
| Sampling Distribution | | | Uniform | | |
| Attention Dropout | | | 0.1 | | |
| MLP Dropout | | | 0.1 | | |
| Residual Dropout | | | 0.1 | | |
| Act Seq Length | 10 | 10 | 10 | 100 | 20 |
| Denoising Steps | 4 | 4 | 4 | 8 | 5 |
| Multistep | 5 | 10 | 10 | 2: Insert 100:Transfer | 15 |
| Camera Views | [Primary Static] | [Primary Static, Wrist] | [Primary Static, Wrist] | [Primary Static] | [Primary Static, Secondary Static] |
| Use Proprio | False | False | False | True | False |
| Action Space | Delta EEF | Delta EEF | Delta EEF | Bi-Joint | Joint |
| Frequency | 3/5 | 10 | 10 | 50 | 6 |

Table 5: **Overview of Hyperparameters used for FLOWER across different Benchmarks**

| **Name** | **Number of Parameters** |
|---|---|
| ViT | 360M |
| VLM | 205M |
| Action Encoders | 3.2M |
| Action Heads | 31.8K |
| Global-AdaLN | 28.3M |
| Cond Linear Proj. | 1.0M |
| Timestep Embedder | 1.3M |
| Cond Norm | 1.0K |
| FreqEmbedder | 1.3M |
| Flow Transformer | 339M |
| Total Parameters FLOWER | 947M |

Table 6: **Overview of Parameter Distribution across all Model Components of FLOWER.**

# A Pretraining Details

Table 7: Dataset Distribution by Percentage

| **Dataset** | **Percentage (%)** |
|---|---|
| bridge_dataset | 28.62 |
| fractal_data | 24.68 |
| droid | 23.50 |
| cmu_play_fusion | 6.15 |
| dobbe | 5.94 |
| libero_10_no_noops | 4.41 |
| libero_goal_no_noops | 4.07 |
| real_kitchen_lang | 2.64 |

We pretrain FLOWER on different datasets mixes that are described in Table 7 using one cluster node with 4 H100 GPUs for 48 hours. We pretrain all variants using a single static image and only use proprioception signal for the bimanual settings. We set the action chunk length to 20 across all settings and condition the model on single image from the current state only for maximum efficiency. Training is conducted using HuggingFace Accelerate for optimized Multi-GPU Training. We created custom PyTorch wrapper for the OXE Torch Dataloaders [6] inspired by efforts from OpenVLA [1]. We train FLOWER using BF-16 accuracy for optimized memory performance. For the Cross-Action-and Delta-EEF Mix we set the batch size to 256 and use 4 gradient accumulate steps to achieve an batch size of 1024. In total we trained for 300k-400k training steps depending on the dataset composition. We did not notice major training instabilities.

## A.1 Pretraining Ablation Experiences

We tested several ideas, that did not work well in our maximum efficiency pretraining settings:

| Hyperparameter | Value |
|---|---|
| GPU Type | H100 |
| N GPUS | 4 |
| Batch Size | 256 |
| Grad accumulate Steps | 4 |
| Optimizer FlowT | AdamW |
| LR Max FlowT | 1e-4 |
| LR Scheduler | warm-up with constant + cosine decay |
| Min LR FlowT | 1e-5 |
| Final LR FlowT | 1e-5 |
| FlowT LR scheduler phases | [0.01, 0.39, 0.6] |
| Max Desired Train Steps | 600,000 |
| Optimizer VLM | AdamW |
| Lr Max VLM | 1e-5 |
| Lr Scheduler | warm-up with constant + cosine decay |
| Min LR VLM | 1e-7 |
| Final LR VLM | 1e-6 |
| VLM LR scheduler phases | [0,1, 0.3, 0.6] |
| EMA | False |
| Weight Decay FlowT | 0.1 |
| Weight Decay VLM | 0.001 |
| Total Training Time | 48 hours |
| Training Steps reached | 350,000 |

Table 8: **Hyperparameteres for Pretraining FLOWER on all Pretraining Data Mixtures**

- **Using variable Length Action Chunks:** We experimented with flexible action chunks during training as done in CrossFormer [41]. However, we noticed slow convergence and lower performance for our SIMPLER experiments. Thus, we decided to use a constant chunk length for all datasets of 20. While many delta EEF datasets like Fractal operate on lowe frequency and Aloha Setups like Biplay [42] require 50 Hz, we find 20 to be a good trade-off that enables easy finetuning to different lengths, thanks to our 1D Rope Position Embeddings.
- **Using Multiple Images with Custom Masking:** We also tested variations in pretraining that involve flexible image padding up to 3 different views depending on the dataset. However, this reduces the overall training speed given the higher number of image tokens and required GPU memory. Also given the more complex setting, we found that the training time of 2 days not enough to achieve convergence. Thus, we limited pretraining to single image. However, our finetuning experiments show the flexbility of FLOWER to adapt to more image views without issues.
- **Mixture-of-Experts Approaches for the Flow Transformer**: We conducted architecture ablation experiments with more action-specialist components. FLOWER ablations used action-type specific MLPs with action-type specific LayerNorms. We also experimented with a shared additional MLP, that all action types use while combining the output with action-specific specialist MLP. However, the training was more memory intensive and had issues with NaN losses. It also showed slower convergence. We found the action-specific Global-AdaLN with all other parameters shared inside the Flow Transformer to work best. However, we believe that future research can address this issue to develop even more efficient architectures for cross-embodiment learning.

### A.2 Language Prompt for the VLA

We encode the meta-information as part of our prompt for the VLM: *"Agent Type: [robot_type], Action Space: [action_space], Task: [task_description]"*.

### A.3 Custom Learning Rate Scheduler

In early experiments with an older version of our model, we observed a substantial drop in performance when training without our custom dual–optimizer learning rate scheduler (4.44 vs 0.8). Upon further investigation after finalizing the model, we found that this was largely due to instabilities in an earlier model variant rather than the absence of the scheduler itself. With our final architecture, replacing the custom scheduler with a standard constant learning rate of $2 \times 10^{-5}$ leads to only a slight reduction in performance, confirming that the scheduler provides only minor benefits rather than being a critical requirement for the performance.

### A.4 Details for Cross-Action Space Flow Transformer

Large transformer models often face stability challenges when simultaneously dealing with different data frequencies and distributions. We address this by allocating individual dual-RMSNorm parameters for each action type, capturing action-specific activation statistics more effectively than a single normalization.

Additionally, we replace standard feed-forward layers with SwiGlu blocks [43], a sandwich-style MLP that we express as follows. For an input vector $\mathbf{h}$,

$$\mathbf{y} \;=\; \Big(\mathrm{Norm}(\mathbf{W}_1\,\mathbf{h})\Big) \;\odot\; \mathrm{SiLU}\Big(\mathrm{Norm}(\mathbf{V}_1\,\mathbf{h})\Big), \tag{3}$$

where $\mathrm{SiLU}(x) = x\,\sigma(x)$, $\mathbf{V}$ and $\mathbf{W}$ are to linear matrices and $\mathrm{Norm}(\cdot)$ indicates RMSNorm in our implementation. Unlike LayerNorm, which subtracts the mean, RMSNorm normalizes each sample by its root-mean-square:

$$\mathrm{RMSNorm}(\mathbf{x}) \;=\; \frac{\mathbf{x}}{\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j^2 \;+\; \epsilon}}, \tag{4}$$

yielding smoother gradients and reduced training instability [26]. Furthermore, we apply QK-value normalization [44] in both self- and cross-attention modules to mitigate large softmax outputs. This additional normalization has been used in many large-scale diffusion and flow transformer architectures in image generation [45] or Diffusion Policies [8].

Together, (i) extended RoPE embeddings, (ii) action-specific encoders/decoders, (iii) a shared AdaLN controller yielding action-specific normalization signals, and (iv) RMSNorm with SwiGLU MLPs enable our Flow Transformer to train stably across heterogeneous data and multiple action spaces. Its modular architecture requires minimal changes when adding new action types, retaining efficient scalability for a wide range of robotic tasks.

## B Detailed Experiments

In this section we provide detailed results and additional comparisons for all simulation environments and real robot experiments.

**CALVIN Benchmark.** [34] A language-conditioned manipulation benchmark containing 24k human-teleoperated demonstrations. Each trajectory spans up to 64 timesteps and encompasses 34 predefined basic skills, including tasks such as "rotate blue block right," "move slider right," "lift red block slider," "turn off light bulb," "open drawer," and "place in drawer." The dataset is divided into four splits (A, B, C, D) and evaluates agents on completing 5 consecutive tasks. For evaluation, an agent must complete a sequence of 5 randomly sampled tasks in order (e.g., "open drawer" $\rightarrow$ "lift blue block drawer" $\rightarrow$ "place in drawer" $\rightarrow$ "close drawer" $\rightarrow$ "turn on light bulb"). The evaluation consists of 1000 rollouts on split D, measuring both the success rate of completing the entire sequence and the average number of successfully completed tasks within each sequence. The Franka Emika Panda robot is controlled via Delta-End-Effector Space with a discrete gripper, utilizing both static and wrist cameras for scene understanding. There are three benchmark types: D$\rightarrow$D, ABC$\rightarrow$D, ABCD$\rightarrow$D, that depend on the used dataset for trainign the policies. After training all are evaluated on the same environment D.

| **Benchmark** | **FLOWER** | **2nd Best** | **Abs. Imp.** | **Rel. Imp. (%)** |
|---|---|---|---|---|
| CALVIN D | 87.0% | 77.0% | 10.0% | 13.0% |
| CALVIN ABC | 90.6% | 85.8% | 4.8% | 5.6% |
| CALVIN ABCD | 93.4% | 90.4% | 3.0% | 3.3% |
| SIMPLER Bridge | 40.0% | 30.0% | 10.0% | 33.3% |
| SIMPLER Google | 32.2% | 42.4% | -10.2% | -24.1% |
| LIBERO OLGS | 96.9% | 97.1% | -0.2% | -0.2% |
| LIBERO 90 | 94.7% | 96.0% | -1.3% | -1.4% |
| Real-World | 61.0% | 30.0% | 31.0% | 103.3% |
| Aloha | 54.0% | 54.0% | 0.0% | 0.0% |
| Real-Generalization | 51.0% | 23.4% | 27.6% | 118.0% |
| **Average** | – | – | 7.5% | 25.1% |

Table 9: Normalized performance improvement of FLOWER compared to its second-best baseline for each benchmark. CALVIN metrics are normalized by dividing the average sequence lengths by 5. Real-Generalization values are computed as the average across Novel Object, Flashlight, BG Distractors, and New Tasks Composition tests. The overall average improvement is computed over all benchmarks.

| Train→Test | Method | PrT | Action Type | VLM | No. Instructions in a Row (1000 chains) | | | | | Avg. Len. |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | 1 | 2 | 3 | 4 | 5 | |
| ABC→D | Diff-P-CNN [39] | × | Diffusion | × | 63.5% | 35.3% | 19.4% | 10.7% | 6.4% | 1.35±0.05 |
| | MDT [37] | × | Diffusion | × | 63.1% | 42.9% | 24.7% | 15.1% | 9.1% | 1.55 |
| | RoboFlamingo [46] | × | Cont. | ✓ | 82.4% | 61.9% | 46.6% | 33.1% | 23.5% | 2.47 |
| | SuSIE [47] | ✓ | Diffusion | × | 87.0% | 69.0% | 49.0% | 38.0% | 26.0% | 2.69 |
| | DeerVLA [22] | × | Cont. | ✓ | 84.8% | 72.3% | 54.9% | 44.6% | 33.5% | 2.90 |
| | GR-1 [48] | ✓ | Cont. | × | 85.4% | 71.2% | 59.6% | 49.7% | 40.1% | 3.06 |
| | OpenVLA [1] | ✓ | Discrete | ✓ | 91.3% | 77.8% | 62.0% | 52.1% | 43.5% | 3.27 |
| | 3DDA [49] | × | Diffusion | × | 93.8% | 80.3% | 66.2% | 53.3% | 41.2% | 3.35 |
| | MoDE [10] | ✓ | Diffusion | × | 96.2% | 88.9% | 81.1% | 71.8% | 63.5% | 4.01±0.04 |
| | RoboDual [17] | ✓ | Diffusion | ✓ | 94.4% | 82.7% | 72.1% | 62.4% | 54.4% | 3.66 |
| | VPP [20] | ✓ | Diffusion | × | 95.7% | 91.2% | 86.3% | 81.0% | 75.0% | 4.29 |
| | Seer [50] | ✓ | Cont. | × | 96.3% | 91.6% | 86.1% | 80.3% | 74.0% | 4.28 |
| | **FLOWER (ours)** | × | Flow | ✓ | **99.3%** | **96.0%** | **90.3%** | **82.3%** | **75.5%** | **4.44±0.04** |
| | **FLOWER (ours)** | ✓ | Flow | ✓ | **99.4%** | **95.8%** | **90.7%** | **84.9%** | **77.8%** | **4.53±0.04** |
| ABCD→D | Diff-P-CNN [39] | × | Diffusion | × | 86.3% | 72.7% | 60.1% | 51.2% | 41.7% | 3.16±0.06 |
| | RoboFlamingo [46] | × | Cont. | ✓ | 96.4% | 89.6% | 82.4% | 74.0% | 66.0% | 4.09 |
| | DeerVLA [22] | × | Cont. | ✓ | 99.1% | 93.3% | 82.1% | 74.6% | 63.8% | 4.13 |
| | GR-1 [48] | ✓ | Cont. | × | 94.9% | 89.6% | 84.4% | 78.9% | 73.1% | 4.21 |
| | MDT [37] | × | Diffusion | × | 98.6% | 95.8% | 91.6% | 86.2% | 80.1% | 4.52±0.02 |
| | MoDE [10] | ✓ | Diffusion | × | 97.1% | 92.5% | 87.9% | 83.5% | 77.9% | 4.39±0.04 |
| | **FLOWER (ours)** | × | Flow | ✓ | **98.9%** | **96.7%** | **93.9%** | **90.2%** | **85.5%** | **4.62±0.03** |
| | **FLOWER (ours)** | ✓ | Flow | ✓ | **99.2%** | **96.9%** | **96.9%** | **92.3%** | **88.3%** | **4.67±0.04** |
| D→D | MDT [37] | × | Diffusion | × | 93.7% | 84.5% | 74.1% | **64.4%** | 55.6% | 3.72±(0.05) |
| | RoboUniView [51] | ✓ | Cont | ✓ | 96.2% | 88.8% | 77.6% | 66.6% | 56.3% | 3.85 |
| | **FLOWER (ours)** | ✓ | Flow | ✓ | **97.4%** | **92.4%** | **86.9%** | **81.3%** | **74.9%** | **4.35±0.02** |

Table 10: **CALVIN Benchmark results for D, ABC and ABCD.** The table reports average success rates for individual tasks within instruction chains and the average rollout length (Avg. Len.) to complete 5 consecutive instructions, based on 1000 chains. Zero standard deviation indicates methods without reported standard deviations.

For all experiments settings, we train FLOWER for up to 40k steps across 4 GPUS with a batch size fo 8 each. The standardized evaluation protocol enable us to directly compare the results of FLOWER against other baselines, which enables a fair comparison to prove the SoTA performance of FLOWER.

**Baselines** We compare FLOWER against a diverse set of available baselines on all CALVIN variants. Relevant VLAs include RoboDual [17], OpenVLA [1], RoboFlamingo [46] and depth-reconstruction pretrained RoboUniView [51]. In addition we consider video-based policies pretrained on diverse video data SeeR [50], Video-Prediction-Policy [20] and GR-1 [48]. Moreover, we consider relevant diffusion-based policies like MoDE [10], MDT [37] and depth-based 3D-Diffusor-

Actor [49] and SuSIE [47]. FLOWER surpasses all of these baselines across all CALVIN variants with remarkable efficiency in just 6 hours of finetuning.

### B.1 SIMPLER Benchmark Tasks

The real2sim benchmark SIMPLER [12] consists of two evaluation challenges, that we describe in detail below:

**Google Robot Setting.** The Google Robot setting comprises four distinct manipulation tasks of varying complexity. The first task, "pick coke can," requires the robot to grasp and lift an empty Coke can from the table. This task includes 75 total trials, testing three different can orientations: horizontally laying, vertically laying, and standing upright. For each orientation, the can is placed at 25 specific grid points within a defined rectangular area on the table, with the environment kept free of distracting elements in its standard configuration.

The second task, "move objects near objects," evaluates the robot's ability to perform relative object positioning with 60 total trials. The setup involves three objects arranged in a triangle pattern, where one object serves as the source, another as the target, and the third as a distractor. The task utilizes eight distinct objects: blue plastic bottle, Pepsi can, orange, 7up can, apple, sponge, Coke can, and Redbull can. Five random triplets are selected from this object pool, with each triplet tested in both upright and inverted triangle patterns. The specific triplet combinations include: (1) blue plastic bottle, Pepsi can, and orange; (2) 7up can, apple, and sponge; (3) Coke can, Redbull can, and apple; (4) sponge, blue plastic bottle, and 7up can; and (5) orange, Pepsi can, and Redbull can.

The third task focuses on drawer manipulation, comprising 54 trials that test the robot's ability to handle articulated objects. The robot is positioned at nine different locations within a rectangular area on the floor and must either open or close a specific drawer (top, middle, or bottom) of a cabinet. This creates a comprehensive evaluation across different robot positions and drawer configurations.

The fourth task combines drawer manipulation with object placement in a 27-trial multi-step interaction. The robot must first open the top drawer and then transfer an apple from the cabinet surface into the drawer. This task evaluates the robot's capability to execute sequential actions, with the robot positioned at three distinct locations and the apple placed at nine specific grid points on the cabinet surface.

**WidowX + Bridge Setting.** The WidowX + Bridge setting features four manipulation tasks, each designed to test different aspects of robotic control. The first task, "put spoon on towel," requires placing a spoon from one corner to another of a 15cm square on the tabletop. The spoon's initial orientation alternates between horizontal and vertical, necessitating appropriate gripper reorientation. This task comprises 24 trials total.

The second task, "put carrot on plate," follows a similar structure to the spoon task but replaces the objects, using a carrot instead of a spoon and a plate instead of a towel. This variation tests the robot's ability to transfer manipulation skills to different objects while maintaining the same spatial constraints.

The third task evaluates precise object stacking, requiring the robot to place a green block (3cm in size) on top of a yellow block. The task includes two square configurations with 10cm and 20cm side lengths, creating different spatial challenges. The blocks are positioned at different corners of these squares, totaling 24 trials.

The final task is "put eggplant into yellow basket,". The eggplant is randomly positioned within the right basin of a sink, while a yellow basket is placed in the left basin. The eggplant's placement varies in both location and orientation but is carefully arranged to remain easily graspable, avoiding proximity to the sink's edges. This task also comprises 24 trials.

We evaluate FLOWER on the SIMPLER benchmark [12] after pretraining on our cross-embodiment mix. SIMPLER is a real2sim benchmark that implements several scenes from the diverse BridgeV2 [32] and Google Robot setup [13] to test foundation policies after pretraining. The benchmark

| Method | Put Carrot on Plate | Spoon on Towel | Stack the Blocks | Eggplant in Yellow Basket | Average |
|---|---|---|---|---|---|
| RT-1-X | 4 | 0 | 0 | 0 | 1.1 |
| Octo | 8 | 12 | 0 | 43 | 16 |
| CrossFormer | 15 | 15 | 0 | **92** | 30 |
| OpenVLA | 0 | 0 | 0 | 4 | 1.0 |
| **FLOWER** | 13 | **71** | **8** | 88 | **45** |

Table 11: **Experimental Results for the SIMPLER Bridge Benchmark.** Average Performance comparison across all Tasks in the Bridge Setting.

| Method | Open/Close Drawer | Move Near | Open Top Drawer and Place Apple | Pick Coke Can | Average |
|---|---|---|---|---|---|
| RT-1-X | **59.7** | 31.7 | **21.3** | **56.7** | **42.4** |
| Octo | 22.7 | 4.2 | 0.0 | 17.0 | 11.0 |
| CrossFormer | 0.5 | 4.6 | 0.0 | 0.0 | 1.3 |
| OpenVLA | 35.6 | **46.2** | 0.0 | 16.3 | 24.5 |
| **FLOWER Cross-X Pret** | 27.8 | 43.3 | 0.0 | **56.3** | 31.9 |

Table 12: **Experimental Results for the SIMPLER Google Robot Benchmark.** Average Performance comparison across different task variations for the Google Robot Setting. All tasks have been tested for Visual Matching and Visual Aggregations Variants and show the average performance across both.

requires policies to run approximately 3000 rollouts in different settings across two benchmarks with 8 different tasks in various conditions.

**Baselines.** On this setup, we compare FLOWER against RT-1X [6], Octo [3], OpenVLA [1], and CrossFormer [41]. For each model, we test on the full benchmark and report the average results for a fair comparison.

**Results.** The results for the Google Robot tasks are summarized in Table 12 and the results for the Bridge challenge are shown in Table 11. Overall, FLOWER outperforms both Octo and OpenVLA on both benchmarks, despite having only 200 GPU hours of pre-training on heterogeneous robot datasets. Notably, the FLOWER variant achieves stronger overall performance, with gains on the Bridge Benchmark. In contrast, on the Google Robot benchmark, RT-1X attains the highest performance across several tasks, suggesting that further improvements in action space modeling or pretraining diversity might be beneficial. However, FLOWER achieves the second best performance and surpasses all other generalist policies in this setting.

These findings show that FLOWER delivers strong performance with low computational demands **(RQ I)** as well as robustly handles diverse robot embodiments and action spaces **(RQ II)**. The robust performance on the Bridge Benchmark highlights its capability to manage diverse action spaces after heterogeneous pretraining, whereas the areas of lower performance on the Google Robot benchmark point to opportunities for further refinement.

### B.1.1 LIBERO Benchmark.

The LIBERO benchmark [35] comprises multiple task suites testing different aspects of robotic manipulation. LIBERO-10 provides 50 demonstrations for 10 tasks, while LIBERO-90 extends to 90 different tasks. Both versions use a Franka Emika Panda robot with end-effector control and dual camera inputs (static and wrist). The benchmark includes five distinct suites: Spatial (testing spatial relationships), Goal (varying objectives), Object (object manipulation), Long (extended task duration), and Suite-90 (diverse short-horizon tasks). Each task is evaluated over 50 trials for each with different starting positions. We finetune FLOWER with 30k training steps on each setting and 50k steps for LIBERO 90 given its increased size.

We compare FLOWER against both generalist policies such as $\pi_0$ [7] and $\pi_0$-FAST [52], the improved $\pi_{0.5}$-ki (knowledge insulation) [53] OpenVLA [1], OpenVLA-OFT [54] Chain-of-Affordance-VLA (CoA-VLA) [21], Octo [3], MiniVLA as well as against the current state-of-the-art specialist policy Baku [40], which uses a small transformer-based model with action chunking. As shown in Table 13, FLOWER significantly outperforms all baselines across every LIBERO variant, achieving near-perfect completion rates with success rates consistently above 93%. Notably,

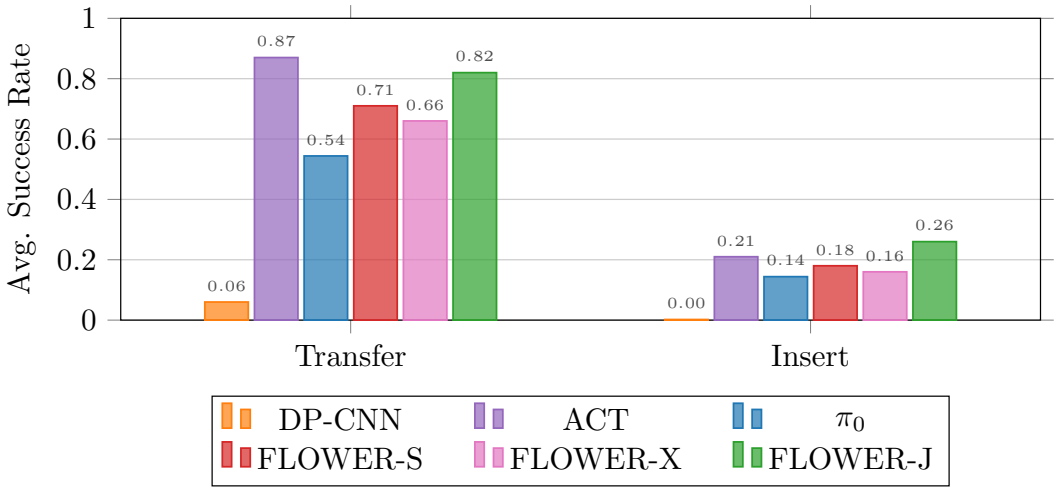



Figure 7: ***Aloha Simulation Tasks:*** Average success rates of different models over 500 evaluations on *Cube Transfer* and *Insertion*. *S* denotes FLOWER trained from scratch, *X* applies cross-action space pretraining, and *J* represents droid-only joint state pretraining.

on LIBERO-Long, FLOWER is the only policy to exceed a 90% success rate (93.5%), while other generalist approaches struggle with these complex, long-horizon tasks (50-54% success rates), with only the specialist Baku model achieving competitive performance in this demanding setting.

| | Spatial SR (↑) | Object SR (↑) | Goal SR (↑) | Long SR (↑) | 90 SR (↑) | Average (without 90) SR (↑) |
|---|---|---|---|---|---|---|
| Diff-P-CNN [39] | 78.3 ± 1.1% | 92.5 ± 0.7% | 68.3 ± 1.2% | 50.5 ± 1.3% | - | 72.4 ± 0.7% |
| Octo [3] | 78.9 ± 1.0% | 85.7 ± 0.9% | 84.6 ± 0.9% | 51.1 ± 1.3% | - | 75.1 ± 0.6% |
| OpenVLA [1] | 84.7 ± 0.9% | 88.4 ± 0.8% | 79.2 ± 1.0% | 53.7 ± 1.3% | - | 76.5 ± 0.6% |
| OpenVLA-OFT [54] | 97.6% | 98.4% | **97.9%** | 94.5% | - | 97.1% |
| CoA-VLA [21] | 85.3 ± 0.9% | 93.1 ± 1.0% | 85.8 ± 0.9% | 55.0 ± 1.2% | - | 79.8 ± 0.5% |
| Baku [40] | - | - | - | 86.0% | 90.0% | - |
| MiniVLA | - | - | - | - | 86.0% | - |
| MoDE [10] | - | - | - | 94.0% | 95.0% | - |
| $\pi_0$ [7] | 96.8% | 98.8% | 95.8% | 85.2% | - | 94.2% |
| $\pi_0$-FAST [52] | 96.4% | 96.8% | 88.6% | 60.2% | - | 85.5% |
| $\pi_{0.5}$-ki (from scratch) [53] | 96.6% | 97.2% | 94.6% | 84.8% | 92.7% | 93.3% |
| $\pi_{0.5}$-ki (from generalist model) [53] | **98.0%** | 97.8% | 95.6% | 85.8% | **96.0%** | 94.3% |
| **FLOWER** | 97.5 ± 0.8% | **99.1 ± 0.4%** | 96.1 ± 0.6% | **94.9 ± 1.2%** | 94.7 ± 1.0% | **96.9 ± 0.7%** |

Table 13: **Experimental Results for the LIBERO Benchmarks.** SR: Success Rate. Best results in each column are shown in bold. FLOWER achieves state-of-the-art results across most tested settings.

### B.1.2 Aloha Benchmark.

The Aloha benchmark [36] provides 50 human-collected demonstrations for two tasks: Cube Transfer and Insertion. In both tasks, a bi-manual Aloha robot operating in joint space is equipped with a single top-view camera and proprioceptive state input. We evaluate each task over 500 episodes, with each episode consisting of 500 steps. For the baselines we adopt ACT[36] and Diffusion Policy[39] implemented by lerobot [55].

Next, we evaluate FLOWER's ability to learn challenging high frequency control on the Aloha simulation setup [36]. We test several versions of FLOWER that have been finetuned on different pretraining mixes: Cross-X and joint state droid only. In addition, we compare FLOWER against two common specialist policies: Diffusion Policies and the state-of-the-art policy for bi-manual setups Action Chunking Tranformer (ACT) [36]. We use the two simulation tasks, "Insert the peg into the socket." and "Pick up the cube with the right arm and transfer it to the left arm.", that are visualized in Figure 4. The dataset contains 50 human-collected demonstrations for each tasks.

**Results.** As shown in Figure 7, FLOWER achieves a strong performance on both tasks with eight denoising steps and outperforms the specialist ACT policy on the challenging Insertion task by a

considerable margin. FLOWER achieves comparable performance on the Transfer task compared to ACT expect for the variant pretrained on cross action space data. The standard Diffusion Policy is not able to solve any of the tasks. Comparing the different pretraining versions of FLOWER, we find that the joint only mix using droid achieves the best results by a considerable margin. Surprisingly, the cross-embodied pretraining is not able to achieve strong results and its final performance is even lower than the version trained from scratch.

### B.2 Real Kitchen Play Dataset.

We conducted data collection through teleoperation, utilizing a leader-follower robot configuration to ensure precision and intuitive control [56]. The dataset includes proprioceptive sensor readings and images captured by two static cameras at a frequency of 6 Hz. Actions were represented as normalized desired joint positions. In total, we curated 417 labeled short-horizon segments, each paired with text instructions. To enhance diversity in task descriptions, GPT-4 was employed to generate varied language annotations.

**Evaluation Protocol.** Each policy is tested 5 times for each task from a starting position not seen in training with some added noise to it. During our experiments, we further varied the orientation of the banana slightly for the robot to pick up, while we kept the toaster in the same position during all our experiments. We report the average success rate and rank for each task to determine the best policy.

#### B.2.1 Pretraining Details for Baselines for the Real World Kitchen

For finetuning the baseline generalist robot policies (Octo, CrossFormer, and OpenVLA), we adhered as closely as possible to the official recommendations provided in their respective GitHub repositories, with only minimal modifications where necessary. All experiments were conducted on a single-node GPU cluster equipped with four NVIDIA RTX 4090 GPUs (24GB each).

For **Octo**, we fine-tuned the model for 50,000 steps, which took approximately 6 hours. In our setup, we used two images per sample—designating the top image as `image_primary` and the side image as `image_secondary`. The baseline Octo model has a default action space of 7 dimensions (delta end-effector position, rotation, and gripper controls). To accommodate our tasks, we extended the default action head by one additional dimension, creating a 7+1 dimensional absolute action space before fine-tuning.

For **CrossFormer**, we again relied on the default fine-tuning settings from the original repository and trained for 50,000 steps, which took around 12 hours. The image setup was identical to that used for Octo. We introduced a new action head, `new_arm_single_joint`, with an action dimension of 8, and developed a new observation tokenizer specifically for the secondary image. All other modules were initialized from the pretrained weights provided in the repository and then fine-tuned.

For **OpenVLA**, which originally supports only delta end-effector actions and enforces an assertion to prevent the use of joint-space OXE datasets for pretraining or fine-tuning, we modified the code to remove this assertion. We then introduced appropriate action and normalization masks (masking nothing for the action and masking the gripper for normalization). Using the default fine-tuning configuration with LoRA-based updates, we trained OpenVLA for 150,000 steps. Due to memory constraints, we reduced the batch size from the default of 16 to 1 and applied gradient accumulation over 4 steps (as opposed to a larger accumulation factor, which would have substantially increased training time). This fine-tuning process took approximately 60 hours on our 4-GPU setup.

For FLOWER we finetuned our model on the kitchen dataset for 50,000 steps. Since FLOWER has been pretrained on single image, we extended the second static image for finetuning. No additional modifications have been made to guarantee a fair comparison against the baselines.

Overall, these modifications allowed us to fine-tune all baseline models under comparable conditions (approximately 100k–150k steps, moderate batch sizes, and consistent GPU resources), ensuring a fair evaluation on our real-world kitchen tasks.

### B.2.2 Failure Cases for different Policies

**Octo.** The most common failure mode involves Octo fixating on the microwave - repeatedly opening, closing, or attempting to interact with its door even when the task involves other objects or locations. The second most frequent failure involves Octo's poor object manipulation, particularly with the pot and banana, where it either drops items prematurely or fails to lift them high enough to clear obstacles like the sink edge. Finally, there's a consistent pattern of spatial navigation issues where Octo either pushes objects into walls, hovers aimlessly above target locations, or places objects in incorrect intermediate positions (like between the sink and stove).

**CrossFormer.** The Crossformer policy exhibits several consistent failure patterns across tasks, including freezing in place, hovering without executing actions, and getting stuck on objects (e.g., sink, microwave door, oven). Many failures involve misinterpreting tasks, such as repeatedly pretending to place toast in the sink or confusing objects like the banana and the oven tray. The model also struggles with manipulating objects correctly, often failing to grasp, dropping, or pushing objects off surfaces rather than placing them accurately. Additionally, it frequently interacts with unintended objects, such as opening and closing the microwave

**OpenVLA.** OpenVLA frequently fails due to object manipulation errors, such as pushing, flipping, or throwing objects off surfaces rather than placing them correctly. A recurring issue is poor grasping ability, especially with pots and bananas, often failing to lift them or dropping them prematurely. Additionally, the policy exhibits random movement behaviors, such as hovering aimlessly, crashing into the kitchen, or moving without executing the task. It also struggles with partial execution, frequently opening and then immediately closing doors or trays instead of completing the full action.

**FLOWER.** The most common failure mode of FLOWER is imprecise spatial positioning, particularly evident in tasks like pushing the toaster lever where the agent consistently misses by about 1cm. We hypothesize that this is due to workspace normalization issues at boundary regions. The second major failure pattern involves interaction with pots in the sink, where FLOWER either gets stuck in loops just before completion, fails to properly clear the sink walls, or incorrectly routes objects (like trying to drop pots into the sink during stove-to-stove transfers). Finally, there are issues with excessive force application in some cases, particularly with the toaster where the agent occasionally rips it off rather than interacting with it properly.

## B.3 Generalization Experiments

Finally, we evaluate FLOWER against the best baseline in several generalization experiments. In these experiments, we test the models under conditions that introduce variations not encountered during training. Table 15 reports the performance of the different methods on tasks such as "Move Pot from Right Stove to Sink", "Open Oven", and "Pull Oven Tray" under three scenarios: Novel Object, Flashlight, and Background Distractors. For instance, in the Novel Object condition, new object instances are introduced, while the Flashlight and Background Distractors settings simulate changes in illumination and environmental clutter. These settings collectively challenge the models to generalize beyond their training distribution.

In particular, our experiments also examine the models' abilities to manipulate novel objects—those not present in the initial training distribution. The objects that are new to the model are highlighted in **bold** in Table 15. As shown, FLOWER consistently achieves higher success rates and lower ranks when manipulating these unfamiliar items, demonstrating robust performance even under significant distribution shifts. Figure 8 provides visual examples of these novel objects and the scene with various background distractions. This additional analysis underscores the strength of our approach in adapting to unseen variations in both object appearance and environmental context.

### B.3.1 Novel Task Compositions

To further evaluate our method's capacity for compositional generalization, we designed a set of novel task compositions that require the agent to combine multiple subtasks into a coherent, long-

horizon plan. Each task is defined as a sequence of actions that must be executed in a specific order. For instance, the **Sequence: Open and Close All Appliances** task comprises the following subtasks: "Open the Microwave", "Open the Oven", "Open the Ice", "Close the Ice", "Close the Oven", and "Close the Microwave". This sequence challenges the model to manipulate various kitchen appliances in a coordinated manner, ensuring that the prescribed order of operations is maintained under varying conditions.

In addition, we introduced two other sequence tasks to test different aspects of compositionality. The **Sequence: Move Items Between Stovetop and Sink** task requires the agent to transfer items between workstations, with subtasks including "Move Banana from Right Stove to Sink", "Push the Toaster Lever", "Move Pot from Left Stove to Right Stove", "Pick Up Toast and Place it at the Sink", "Move Pot from Right Stove to Left Stove", and "Move Banana from Sink to Right Stove". Finally, the **Sequence: Operate the Oven** task focuses on oven manipulation and is composed of the subtasks "Open the Oven", "Pull the Oven Tray", "Move Banana from Right Stove to Oven Tray", "Push the Oven Tray", and "Close the Oven". These novel task compositions simulate realistic, multi-step scenarios and provide a rigorous benchmark for evaluating the ability of our approach to integrate learned sub-skills into coherent, long-horizon behaviors.

(a) Unseen Objects for Generalization Experiments

(b) Cluttered Scene with Distractors

Figure 8: **Generalization Experiments:** Examples of unseen objects (left) and cluttered scenes (right) used to test the adaptability of the policies in our real-world setting. All the tested objects are not included in the training dataset.

## C Extended Related Work

**Cross-Embodiment Learning** A core challenge in robotics is learning a unified policy for heterogeneous embodiments with distinct action and sensor spaces. Early approaches often applied modular policies [57] or hardware-conditioned representations [58], and some leveraged graph-based representations to generalize across different robot hands [59, 60]. However, these efforts tended to focus on smaller datasets or simplified environments. Recent work on large-scale cross-embodiment includes RoboCat [14] and PolyBoT [60], which use action tokenization or hierarchical controllers, respectively. Liu et al. [8] propose a unified 258-dimensional action space (RDT-1B) with fixed action prediction length of 64 for all action spaces, while CrossFormer [41] employs separate action heads with a continuous action prediction head for different embodiments but lacks a pretrained vision-language component for generalization to diverse instructions. By incorporating an action-type Global AdaLN conditioned Flow Transformer with a pretrained VLM, FLOWER efficiently handles multiple embodiments while maintaining both action expressiveness and semantic understanding.

**Rectified Flow and Diffusion Models in Robotics.** Diffusion models [61, 62, 63] have become widely used for generating continuous robot actions from visual inputs [39, 3, 64, 38], offering multi-modal behavior [65] and good scaling with large datasets [3, 8, 10]. More recently, *Rectified Flow* [45, 66, 28] has emerged as a promising alternative, enabling a straight-line probability path for

| Task | SR/R | Octo | OpenVLA | CrossFormer | FLOWER |
|---|---|---|---|---|---|
| Pot from right stove to sink | SR | 0.0% | 60.0% | **100.0%** | 80.0% |
| | R | 4 | 3 | **1** | 2 |
| Pot from sink to right stove | SR | 0.0% | 0.0% | 0.0% | **20.0%** |
| | R | 2 | 2 | 2 | **1** |
| Open oven | SR | 40.0% | 0.0% | 0.0% | **60.0%** |
| | R | 2 | 3 | 3 | **1** |
| Pull oven tray | SR | 0.0% | 40.0% | 0.0% | **100.0%** |
| | R | 3 | 2 | 3 | **1** |
| Open microwave | SR | 40.0% | **100.0%** | 40.0% | **100.0%** |
| | R | 3 | **1** | 3 | **1** |
| Close microwave | SR | 20.0% | 80.0% | 40.0% | **100.0%** |
| | R | 4 | 2 | 3 | **1** |
| Banana from right stove to sink | SR | 10.0% | 40.0% | 40.0% | **100.0%** |
| | R | 4 | 2 | 2 | **1** |
| Banana from sink to right stove | SR | 0.0% | 0.0% | 20.0% | **60.0%** |
| | R | 3 | 3 | 2 | **1** |
| Push toaster lever | SR | 0.0% | **100.0%** | 0.0% | 0.0% |
| | R | 2 | **1** | 2 | 2 |
| Pickup toast and put to sink | SR | 0.0% | 20.0% | **80.0%** | 40.0% |
| | R | 4 | 3 | **1** | 2 |
| Open Ice | SR | 0.0% | 0.0% | 0.0% | **100.0%** |
| | R | 2 | 2 | 2 | **1** |
| Banana from right stove to oven tray | SR | 0.0% | 0.0% | 0.0% | **40.0%** |
| | R | 2 | 2 | 2 | **1** |
| Pot from sink to left stove | SR | **0.0%** | **0.0%** | **0.0%** | **0.0%** |
| | R | **1** | **1** | **1** | **1** |
| Pot from left stove to right stove | SR | 40.0% | 0.0% | 20.0% | **80.0%** |
| | R | 2 | 4 | 3 | **1** |
| Banana from tray to right stove | SR | **0.0%** | **0.0%** | **0.0%** | **0.0%** |
| | R | **1** | **1** | **1** | **1** |
| Close oven | SR | 40.0% | **100.0%** | 20.0% | 80.0% |
| | R | 3 | **1** | 4 | 2 |
| Pot from right stove to left stove | SR | 30.0% | 0.0% | 0.0% | **40.0%** |
| | R | 2 | 3 | 3 | **1** |
| Push oven tray | SR | 0.0% | 20.0% | **40.0%** | 20.0% |
| | R | 4 | 2 | **1** | 2 |
| Pot from left stove to sink | SR | 0.0% | 80.0% | 40.0% | **100.0%** |
| | R | 4 | 2 | 3 | **1** |
| Close Ice | SR | 0.0% | 0.0% | 0.0% | **100.0%** |
| | R | 2.0 | 2 | 2 | **1** |
| **Overall Performance** | SR | 10% | 31% | 22% | **61%** |
| | R | 2.70 | 2.10 | 2.20 | **1.25** |

Table 14: **Detailed Results for all tested Real Robot Tasks in the Kitchen Environment.** Each task has two rows: the first (SR) reports success rate (%), and the second (R) reports rank within that task (lower rank = better performance). The best results per task are highlighted in **bold**.

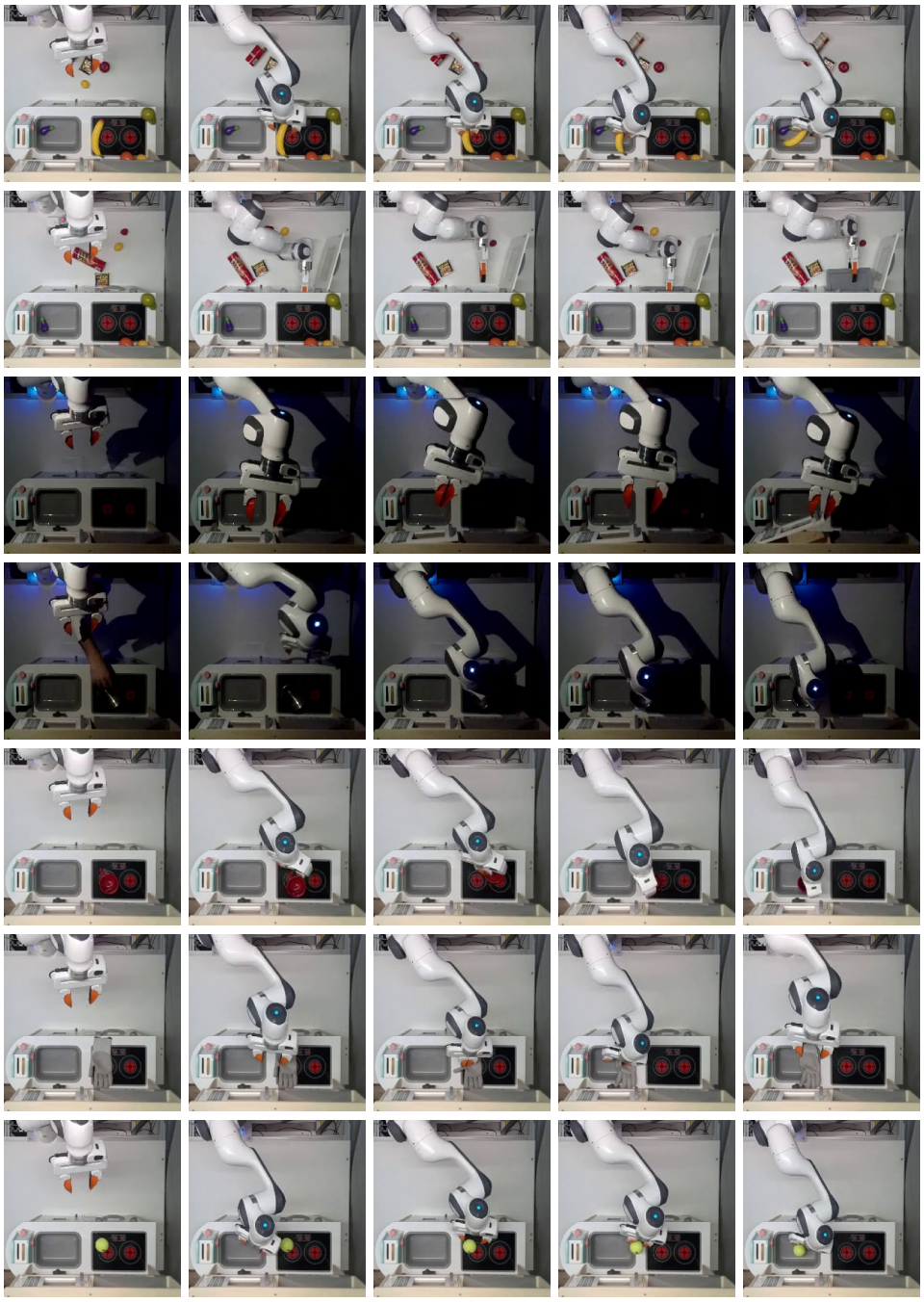


Figure 9: **Example generalization rollouts.** First two rows show rollouts with background distractors, rows 3 and 4 show rollouts with only a flashlight as a light source, and the last 3 rows showcase novel objects.

| Task | Novel Object | | Flashlight | | BG Distractors | |
|---|---|---|---|---|---|---|
| | FLOWER | OpenVLA | FLOWER | OpenVLA | FLOWER | OpenVLA |
| Move the **black donut** from sink to right stove | **33.3** | 0.0 | - | - | - | - |
| Move the **tennis ball** from right stove to sink | **66.7** | 33.3 | - | - | - | - |
| Move the **tennis ball** from sink to right stove | 0.0 | 0.0 | - | - | - | - |
| Move the **black donut** from right stove to sink | **33.3** | 0.0 | - | - | - | - |
| Move the **red cup** from right stove to sink | 33.3 | **66.7** | - | - | - | - |
| Move the **glove** from sink to right stove | **33.3** | 0.0 | - | - | - | - |
| Move the **carrot** from right stove to sink | **33.3** | 0.0 | - | - | - | - |
| Move the **glove** from right stove to sink | **100** | 0.0 | - | - | - | - |
| Move the **carrot** from sink to right stove | 0.0 | 0.0 | - | - | - | - |
| Move the **red cup** from sink to right stove | 0.0 | 0.0 | - | - | - | - |
| Open the microwave | - | - | **100** | **100** | **100** | **100** |
| Pull the oven tray | - | - | **100** | 0.0 | **100** | 66.7 |
| Move banana from right stove to sink | - | - | **100** | 33.3 | 66.7 | 66.7 |
| Close the oven | - | - | 33.3 | **66.7** | 66.7 | 0.0 |
| Push down the toaster lever | - | - | 0.0 | 0.0 | 33.3 | **100** |
| Move pot from right stove to sink | - | - | - | 0.0 | **66.7** | 33.3 |
| Open the ice box | - | - | 0.0 | 0.0 | 66.7 | 0.0 |
| Open the oven | - | - | **100** | 0.0 | **100** | 0.0 |
| Close the microwave | - | - | **100** | **100** | **100** | 66.7 |
| Move pot from left stove to sink | - | - | 0.0 | 0.0 | 33.3 | 66.7 |
| Close the ice box | - | - | 0.0 | 0.0 | **100** | 0.0 |
| Pick up toast and put it in the sink | - | - | 0.0 | 0.0 | 0.0 | 0.0 |
| **Average** | **33.3** | 10.0 | **50.0** | 25.0 | **69.5** | 41.7 |

Table 15: **Generalization experimental results for novel objects, distractions and new lighting conditions.** The table reports the success rate (in %) of the corresponding policy evaluated under three different generalization scenarios: Novel Object, Flashlight, and Background Distractors (evaluated 3 times for each setting). The best score for each test is highlighted in **bold**. A dash (-) indicates that the task was not evaluated in that scenario.

| Task | Method | 1 | 2 | 3 | 4 | 5 | Avg. Seq. Len. |
|---|---|---|---|---|---|---|---|
| Seq: Stovetop + Sink | FLOWER | **66.7%** | **66.7%** | **66.7%** | **33.3%** | **33.3%** | **2.67** |
| | OpenVLA | **66.7%** | 33.3% | 33.3% | 0.0% | – | 1.33 |
| Seq: Open Close All | FLOWER | **100%** | **100%** | **100%** | **100%** | **100%** | **5.00** |
| | OpenVLA | 33.3% | 0.0% | – | – | – | 0.33 |
| Seq: Oven | FLOWER | 0.0% | – | – | – | – | 0.00 |
| | OpenVLA | 0.0% | – | – | – | – | 0.00 |
| **Overall Performance (FLOWER)** | | **51.1%** | — | — | — | — | **2.56** |
| **Overall Performance (OpenVLA)** | | **16.7%** | — | — | — | — | **0.55** |

Table 16: **Long Horizon Task Composition Results.** For each sequence task, the per-instruction success rates (in %) are shown for the first 5 instructions (if applicable) along with the average sequence length. "–" indicates that no instruction was successfully solved at that index. The Overall Performance rows report the average success rate (computed over all available instructions) and the average sequence length across tasks for each method.

action sampling that requires few discretization steps. In robotic policy learning, ActionFlow [67], $\pi_0$ [7] and others [68, 69] showed rectified flow can generate actions more rapidly than standard diffusion. Such fast inference is crucial for high-frequency robot setups like Aloha [70]. Yet, these works are typically confined to a single embodiment or a single action space. By contrast, FLOWER is the first completely open-source policy to apply rectified flow as a *generalist* policy component, unifying expressive and multimodal flow-based action generation with diverse vision-language contexts. While $\pi_0$ published their weights for finetuning, the code for pretraining and their dataset remains closed source.