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 (IRL), Karlsruhe Institute of Technology, Germany
[2]Microsoft Research

ABSTRACT

This work introduces FLOWER, an efficient, open-source Vision-Language-Action Flow policy. Vision-Language-Action (VLA) models have demonstrated remarkable potential for language-guided robotic manipulation by leveraging large-scale vision-language pretraining. However, existing approaches often rely on multi-billion-parameter architectures and massive datasets, making them prohibitively expensive to train. FLOWER is a novel generalist policy that not only outperforms current VLAs but also substantially lowers the computational burden for pretraining, fine-tuning, and inference. FLOWER combines a Rectified Flow Policy with a compact Vision-Language Model (VLM) backbone. The Flow Policy enables expressive, multimodal action generation. The compact VLM backbone provides robust semantic grounding while requiring only a fraction of the usual compute cost. Experiments across 4 simulated benchmarks and real-world settings on more than 100 tasks reveal that FLOWER consistently surpasses foundation policies, e.g., OpenVLA. FLOWER achieves superior performance while significantly reducing both training time and memory requirements. Both the performance and the training efficiency are maintained across different action spaces, showcasing the potential of FLOWER to handle diverse control tasks with affordable deployment, fine-tuning and customization. To encourage further research and the democratization of pretrained VLAs, we open-source the full pretraining and fine-tuning code along with the trained weights.

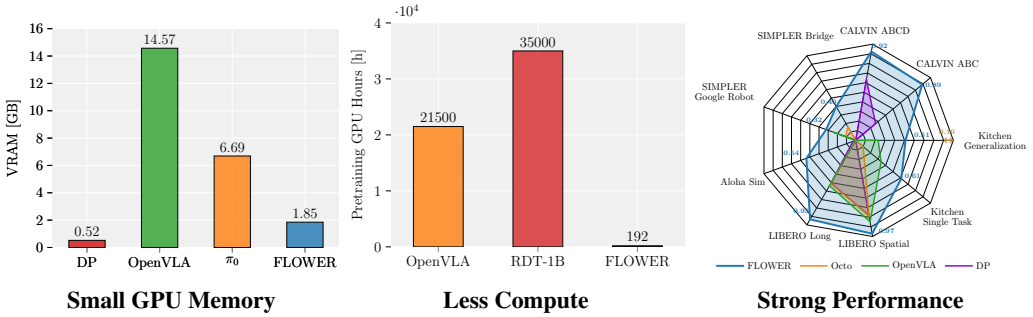


Figure 1: **We present FLOWER, a novel and efficient VLA-Flow Policy with less than 1B parameters.** It achieves SOTA performance across various benchmarks in simulated and real setups across 3 different action spaces and 4 embodiments. It achieves these results with 1% of pretraining GPU hours compared to recent VLAs like OpenVLA Kim et al. (2024).

1 INTRODUCTION

The quest to develop generalist robotic policies that can execute a wide variety of tasks across different embodiments has long been an overarching goal in robotics. Recent advances in imitation learning have made significant progress toward this vision, particularly along two research avenues:

Diffusion Policies (Chi et al., 2023; Reuss et al., 2023) and Vision-Language-Action-Model (VLA) Policies (Brohan et al., 2022; Kim et al., 2024; Octo Model Team et al., 2023).

Diffusion Policies leverage conditional generative models to iteratively denoise action sequences from random Gaussian noise. Recent work has demonstrated success in scaling these models to larger datasets (Liu et al., 2024c; Reuss et al., 2024a). Subsequently, Flow-based methods (Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022) have emerged as competitive alternatives, offering fewer denoising steps and lower computational overhead (Esser et al., 2024; Funk et al., 2024). Unfortunately, their semantic grounding capabilities are quite limited, particularly when pretrained language encoders fail to bridge vision-action contexts. However, these grounding capabilities are crucial for generalizing learned behavior to unseen object scenes and tasks. Reaching these grounding abilities with currently available robotics dataset (Collaboration, 2023) appears infeasible, since the amount of language-annotated trajectories is still relatively small compared to vision-language datasets (Schuhmann et al., 2022; Laurençon et al., 2022).

One line of policies that address this issue are VLAs. VLAs utilize the semantic power of pretrained Vision-Language-Models (VLMs). To adopt VLMs for policy learning, they are fine-tuned to predict discrete or continuous robot actions (Li et al., 2024a; Kim et al., 2024; Driess et al., 2023; Collaboration, 2023). However, state-of-the-art VLA policies like OpenVLA (Kim et al., 2024) or RoboFlamingo (Wu et al., 2024) commonly contain billions of parameters, demanding large-scale GPU resources for both pretraining, fine-tuning and deployment on real robot hardware. This extreme size makes it also challenging to adopt them for high frequency control (Zhao et al., 2023).

To achieve better generalist policies we require new hybrid approaches, that combine expressive, multimodal action generation with linguistic grounding in an efficient way. Hence, we present **Flo**rence **W**ith **E**mbodied Flow (FLOWER), a novel and efficient VLA-Flow policy that achieves state-of-the-art performance across 6 manipulation benchmarks in simulation and real world across diverse robotic tasks while significantly reducing pretraining, fine-tuning, and inference costs.

FLOWER combines the expressive and multimodal Action Generation of Flow-based Policies with the semantic grounding and internet-scale pretraining of VLM. FLOWERs novel hybrid architecture design enables effective pretraining on heterogeneous robotic datasets with less than 200 H100 GPU hours. FLOWER makes the following contributions:

(1) A Novel Hybrid Model Combining a Compact VLM with an Expressive Flow Transformer. FLOWER integrates a compact VLM backbone(Xiao et al., 2024) with an expressive Flow Transformer (Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022). FLOWER combines semantic grounding and expressive, multimodal action generation while requiring $1B$ parameters in total. To the best of our knowledge this makes FLOWER the smallest VLA available.

(2) Low-Compute Pretraining with Wide Accessibility. The pretraining of VLA models typically requires extreme computing cost, e.g., 20k GPU hours or more (Kim et al., 2024; Liu et al., 2024c). In stark contrast, FLOWER surpasses current state-of-the-art VLA results after merely 200 H100 Training hours of pretraining, which represents 1% of the OpenVLA pretraining compute.

(3) Efficient Cross-Action Space Conditioning. FLOWER introduces the action-specific Global-AdaLN Conditioning for Flow Transformers, leveraging action-specific conditioning to optimize parameters without compromising performance. This design ensures seamless adaptation to various robot embodiments, including bimanual manipulation, while maintaining parameter and training efficiency.

We verify these contributions across 190 different tasks in 5 simulations and 1 real world setting. Across 8 benchmarks spanning diverse simulation and real-world scenarios, including experiments on four distinct robot embodiments and three different action spaces, FLOWER demonstrates an average performance improvement of 32.7% compared to current state-of-the-art specialist and generalist policies (see Table 8 for details).

2 Related Work

Early work on large-scale Imitation Learning (IL) (Osa et al., 2018) explored how increasing the number of trajectories impacts policy performance, with Pinto & Gupta (2016) investigating datasets of up to 600k demonstrations. More recently, Open-X-Embodiment (OXE) (Collaboration, 2023)

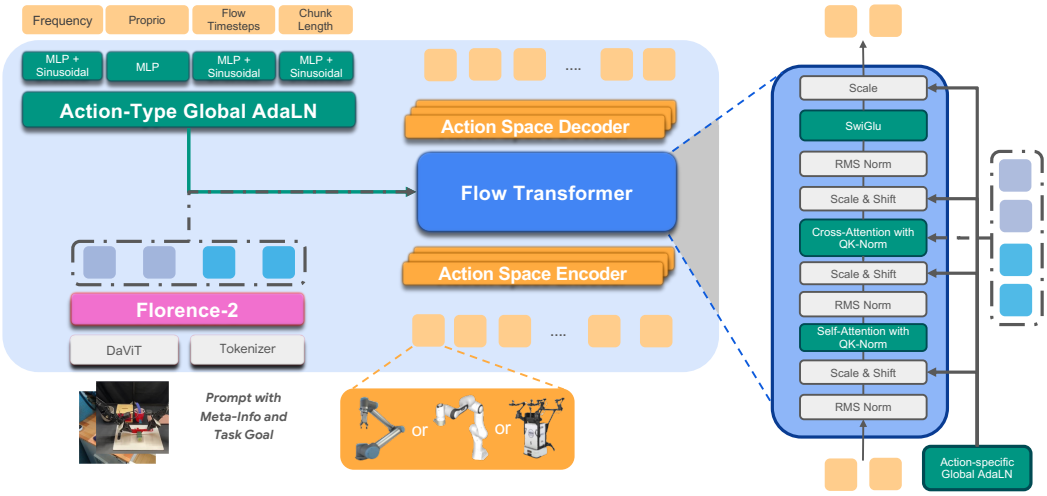


Figure 2: **Detailed Overview of proposed architecture of FLOWER.** First, language and camera views are encoded using a fine-tuned Florence-2 VLM. In this process, images are encoded by the DaViT image encoder, and language and text tokens are processed jointly. Next, the conditioning information is injected into the Flow Prediction Transformer via cross-attention. Finally, our Flow Transformer predicts the velocity field for action generation. FLOWER employs action-specific Global AdaLN-Zero conditioning with meta-information such as embodiment frequency and the current flow process time to efficiently guide action generation.

introduced 1.4M trajectories spanning more than 20 robot embodiments, enabling research into generalist policies for heterogeneous robots.

Several methods build on top of OXE and aim to learn such generalist policies. Octo (Octo Model Team et al., 2023) trains a transformer-based Diffusion Policy on a subset of OXE and only focuses on delta end-effector actions. It does not leverage any pretrained vision-language encoders and relies on training all model components from scratch. This limits Octo's ability to generalize as observed e. g. in benchmarks like SIMPLER (Li et al., 2024b). In contrast, RDT (Liu et al., 2024c) incorporates a large Diffusion transformer with 1.2B learnable parameters plus 11.4B parameters in pretrained vision and language encoders. Although RDT unifies all actions and proprioception into a 258-dimensional latent space, it is computationally expensive (one month of pretraining on 48 A100 GPUs) and challenging to fine-tune because of its large unified action space. These challenges highlight the need for a more efficient generalist policy. RoboDual (Bu et al., 2024) tries to combine VLM's with diffusion policies while using asynchronous updates. It uses a combination of OpenVLA and a small Diffusion Transformer, that is learned on the local robot domain. Latent Bridge (Shentu et al., 2024) outputs in a slower update frequency high level commands from a finetuned VLM, that is used to conditioned a Diffusion policy instead of using simple latent text embeddings. Both approaches have the same disadvantages of having a specialist policy that lack generalist pretraining while still using very large VLM that are very compute and memory intensive to finetune. FLOWER combines the best of both approaches in a single unified, generalist policy.

Several other approaches directly integrate VLM for policy learning. OpenVLA (Kim et al., 2024) fine-tunes a VLM for action prediction but is limited to discrete single delta end-effector actions, making it less suitable for high-frequency control settings. In addition, given its size of 7.7B parameters, it is very resource intensive to finetune and deploy on real robot setups. Similarly, RoboFlamingo (Wu et al., 2024) introduces alternative VLAs, that use continuous action head predictions instead of discrete ones. While its proposed models are smaller than OpenVLA with 3B parameters, they are still expensive to pretrain and deploy. In order to reduce size, a lightweight diffusion head on top of smaller VLM backbones (Wen et al., 2024) can be added or discrete action token approaches can be predicted, e. g. RT-1 (Brohan et al., 2022) and RT-2 (Brohan et al., 2023). GR-1 (Wu et al., 2024) and GR-2 Cheang et al. (2024) combine a GPT-style transformer with video prediction objectives for pretraining generalist VLA policies. Similarly, future frame generation models were combined with action prediction from various others (Tian et al., 2024; Hu et al., 2024;

Reuss et al., 2024b). Meanwhile, HPT (Wang et al., 2024) scales up a transformer backbone with different projectors for each action space but trains from scratch without any language understanding. Beyond learning the model, choosing the correct data sources (Liu et al., 2024b; Lin et al., 2024; Hejna et al., 2024) as well as autonomous data collection and labeling (Blank et al., 2024; Mandlekar et al., 2023) are both key aspect of generalist policy pretraining. Most related to our approach is π_0 Black et al. (2024), which also uses a pretrained VLM for flow-based action generation. π_0 applies an early fusion approach to bridge the modality gap between VLM weights and Flow Prediction. In contrast, FLOWER is the first VLA to introduce intermediate modality fusion, where we conditioned our Flow Transformer on intermediate VLM tokens and cut off a big set of the VLM layers for increased efficiency.

3 Preliminaries and Problem Formulation

In this section, we summarize the background and problem statement to clarify the objective of learning a *generalist* policy from heterogeneous, language-annotated robot trajectories.

3.1 Problem Setup

Learning a generalist policy across diverse robot embodiments poses significant challenges due to heterogeneity in action spaces (such as variations in degrees-of-freedom, control modes like joint vs. end-effector, and actuation frequencies), observation spaces (including differences in sensor suites like the number of camera configurations and proprioceptive sensing), and task specifications (such as variability in language goal grounding across embodiments).

Given a dataset $\mathcal{D} = \{\boldsymbol{\tau}_i\}_{i=1}^{K}$ containing demonstrations from M distinct robot embodiments, where each trajectory $\boldsymbol{\tau}_i = \{(\bar{\boldsymbol{s}}_n, \bar{\boldsymbol{a}}_{n,k}, \boldsymbol{g}_n, \boldsymbol{e}_i)\}_{n=1}^{N}$ contains the current state $\bar{\boldsymbol{s}}_n \in \mathbb{R}^{d_s}$, the action sequence $\bar{\boldsymbol{a}}_{n,k} \in \mathbb{R}^{d_a \times k}$ of length k starting at timestep n, the natural language instruction specifying the sub-task goal $\boldsymbol{g}_n$, and the language-based embodiment meta-description $\boldsymbol{e}_i$, e.g., *"Robot: Franka Panda, Action Space: Delta end-effector"*. Our objective is to learn a goal-conditioned policy $\pi\left(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g}\right)$ that maps from the current observation and embodiment specification to action sequences

$$\pi\left(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g}\right) : (\bar{\boldsymbol{s}}_n, \boldsymbol{g}_n, \boldsymbol{e}_i) \mapsto \bar{\boldsymbol{a}}_{n,k}.$$

The training objective maximizes the action prediction likelihood across all embodiments

$$\mathcal{L}_{\text{IL}} = \mathbb{E}_{(\bar{\boldsymbol{s}}, \bar{\boldsymbol{a}}, \boldsymbol{g}, \boldsymbol{e}) \sim \mathcal{D}} \left[\log \pi_\theta\left(\bar{\boldsymbol{a}}|\bar{\boldsymbol{s}}, \boldsymbol{g}, \boldsymbol{e}\right)\right],$$

where the expectation is over state-action-goal-embodiment tuples sampled from the large-scale heterogeneous robot dataset. This formulation enables learning a single policy that adapts its behavior based on both the current task goal ($\boldsymbol{g}$) and physical embodiment constraints ($\boldsymbol{e}$).

4 Method Overview

4.1 Rectified Flow for Action Generation

At the core of FLOWER's architecture lies the Rectified Flow Transformer for continuous action prediction. Rectified Flow is a scalable and expressive generative modeling framework that requires relatively few denoising steps for expressive multimodal action generation. Rectified Flow models iteratively refine a sample from random Gaussian noise. In contrast to diffusion models, Rectified Flows ensure that the interpolation paths between noise and data are straight-line velocity fields. This streamlined approach reduces the computational burden of inference while preserving expressiveness, making it particularly well-suited for robotics applications where latency is crucial.

Given the conditional action distribution $\pi_\theta\left(\bar{\boldsymbol{a}}_{n,k}|\bar{\boldsymbol{s}}_n, \boldsymbol{g}, \boldsymbol{e}\right)$, the trajectory interpolation between action sequences and noise is modeled through a rectified flow process

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

where $t \in [0, 1]$ represents the normalized flow time step, $\bar{\boldsymbol{a}}_{n,k} \in \mathbb{R}^{d_a}$ is the ground truth action sequence starting at timestep n, and z_1 is standard Gaussian noise matching the action sequence

dimension. The flow time t is sampled from a logit-normal distribution $t \sim \sigma(\mathcal{N}(0, I))$. The model then learns to predict the velocity field between noise and actions by optimizing

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

where v_θ is the rectified flow model underlying the policy π_θ. The flow model v_θ that is conditioned on the current state $\bar{\boldsymbol{s}}$, language goal $\boldsymbol{g}$, and embodiment information $\boldsymbol{e}$.

During inference, the action sequences are generate through progressive denoising

$$\begin{aligned} z_0 &\sim \mathcal{N}(0, I) \\ z_{t+\Delta t} &= z_t - \Delta t \cdot v_\theta(z_t, t, \bar{\boldsymbol{s}}_n, \boldsymbol{g}, \boldsymbol{e}), \end{aligned}$$

where $\Delta t = 1/N$ is the step size for N sampling steps. Our experiments demonstrate strong results with just $N = 4$ denoising steps in single arm settings and $N = 8$ for high frequency dual arm settings. By integrating Rectified Flow into FLOWER, we ensure an expressive and highly efficient policy architecture.

4.2 Leveraging Internet-scale Data for Generalist Policy Learning

Existing VLA models often use large Large-Language-Models (LLMs), such as LLama2-7B in OpenVLA (Kim et al., 2024). They also rely on large Vision Transformer (ViT)-based encoders like SigLip (Zhai et al., 2023) that produce long vision token sequences, making them prohibitively expensive to train or deploy. Instead, FLOWER adopts Florence-2-Large (Xiao et al., 2024) for three main reasons. First, Florence-2 provides robust multimodal embeddings (images and text) while being smaller than many popular VLMs models, with just 770M parameters for ViT (360m) and LLM (410M) combined. Second, it is pretrained on a diverse dataset comprising 5B text-image pairs, employing an efficient Dual-Attention Vision Transformer (DaViT) that generates fewer tokens than typical ViT-based models. Third, Florence2 features an encoder-decoder architecture with an about equal number of parameters for encoder and decoder. While the decoder specializes in next-token prediction, the encoder simply compresses the inputs to general latent features. Additionally, empirical studies on LLM interpretability (Gao et al., 2024) suggest that middle-layer features are often more robust for downstream tasks than final-layer (decoder) outputs. Thus, FLOWER discards the decoder of Florence and directly leverages the intermediate encoder features to condition the Flow Transformer. This choice reduces the number of parameters by 205M and improves training and inference speed without affecting performance (cf. Section 5.6). Florence-2-Large with reduced number of layers provides a compact and computationally accessible VLM with just 565M parameters, enabling FLOWER to remain efficient and scalable while delivering multimodal feature representations. To the best of our knowledge, this makes FLOWER the smallest open-source VLA policy, that leverages internet scale pretraining.

Training the pretrained VLMs jointly with randomly initialized Flow modules presents significant optimization challenges. To address these challenges, FLOWER deploys separate optimizers with distinct learning rates for the Flow Transformer and the VLM. The VLM optimizer leverages a lower warm-up rate to prevent overshooting. Additionally, we implement gradual fine-tuning, starting the VLM at a minimal learning rate to avoid overwhelming the newly initialized Flow layers. These strategies collectively stabilize training and enable the convergence of our 1B-parameter system without requiring excessive computational resources. Ablation studies demonstrate that using a single shared optimizer can degrade benchmark performance by up to 80% (see Section 5.6).

4.3 Cross-Action Space Flow Transformer

A detailed overview of the Flow Transformer components is shown in Figure 2. The conditional information of the current state and goal are injected using Cross-Attention in each block. The intermediate output tokens from Florence2 LLM are projected using a linear layer and RMSNorm (Zhang & Sennrich, 2019). Our Flow Transformer addresses the challenge of dealing with training on heterogeneous robotics data across different action spaces. Below, we detail the core components that tackle these challenges efficiently and scalable.

1) Action-Specific Encoders and Decoders. Each action type (e. g., delta-EEF vs. joint angle) utilizes a small MLP-based encoder/decoder integrated with a shared Flow Transformer core. This

Figure 3: **Evaluated simulation environments.** From left to right: Fractal based on Google dataset (Li et al., 2024b), Berkley Bridge V2 from SIMPLER (Li et al., 2024b), LIBERO (Liu et al., 2024a), CALVIN (Mees et al., 2022) and Aloha Simulation Benchmark (Zhao et al., 2023).

design maintains weight sharing for efficiency while adapting to dimension-specific statistics, as demonstrated in prior work (Doshi et al., 2024; Liu et al., 2024c).

2) Action-Specific Global AdaLN-Zero. To inject global conditioning signals (e. g. time steps, frequencies and action chunk length), we use a modified adaptive layer normalization (AdaLN) (Perez et al., 2018). Standard AdaLN-Zero assigns unique parameters to each layer, resulting in a significant amount of additional parameters, 35% of all parameters in Diffusion Transformers (Peebles & Xie, 2023). The Zero suffix refers to the initialization of the final output layer of the modulation with zeros. To maintain a reduced number of parameters we introduce action-specific global AdaLN-Zero. This approach generates distinct modulation signals for each action type, shared across all layers. This significantly reduces parameters while maintaining performance.

3) Variable Action-Sequence Length. To handle varying action-sequence lengths across robot embodiments, e.g., Aloha requiring 50-100 steps, we leverage *1D Rotary Positional Encoding (RoPE)* (Su et al., 2021). Unlike standard absolute or learned relative encodings, RoPE extends up to length 200 without large embeddings or rigid positions. Specifically, for a token embedding $\mathbf{x}_p \in \mathbb{R}^d$ at position p, we partition its dimension into pairs $(2i,\, 2i+1)$. The modified embedding contains the corresponding pairs

$$
\begin{aligned}
x'_{p,2i} &= x_{p,2i}\cos(\theta_p) \;-\; x_{p,2i+1}\sin(\theta_p), \\
x'_{p,2i+1} &= x_{p,2i}\,\sin(\theta_p) \;+\; x_{p,2i+1}\cos(\theta_p),
\end{aligned}
$$

with $\theta_p = \frac{p}{10000^{2i/d}}$. This rotation "mixes" the two components of the pair in a way that depends on the token's position, thereby injecting positional information into the embedding without adding extra parameters.

4.4 Training Setup

Unlike prior VLA models that utilize up to 50 datasets, FLOWER focuses on a small, curated selection of datasets. This smaller collection emphasizes broad environmental diversity, which is vital for generalization (Gao et al., 2024; Liu et al., 2024b).

We evaluate three variants of pretraining: Cross-Action-Mix, Delta-EEF-Mix and Joint-State-Droid-Only Mix. A detailed breakdown of the data split and action space distribution is provided in Appendix B. In total, our dataset for Cross-Action contains approx. $250,000$ trajectories across 9 datasets, where 64% are delta-EEF data, 21% single arm joint state and 15% bi-manual trajectories. These datasets cover a range of the most commonly used robotic embodiments, including Franka Pandas, Alohas, and XARM robots. Our delta-EEF mix covers different setups with 6 different datasets from 3 embodiments. For all pretraining we set the action chunking length to 20. We also restrict the number of images during pretraining to a single static image. Proprioception data is only used for bi-manual datasets. By pretraining on these mixture, FLOWER captures a wide spectrum of task-relevant features while retaining computational efficiency to achieve strong results with 200 GPU hours pretraining time. For details regarding pretraining mix, hyperparameters we refer to Appendix B in the Appendix.

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 (Chi et al., 2023)	×	Diffusion	×	63.5%	35.3%	19.4%	10.7%	6.4%	1.35±0.05
	MDT (Reuss et al., 2024b)	×	Diffusion	×	63.1%	42.9%	24.7%	15.1%	9.1%	1.55
	RoboFlamingo (Li et al., 2023)	×	Cont.	✓	82.4%	61.9%	46.6%	33.1%	23.5%	2.47
	GR-1 (Wu et al., 2024)	✓	Cont.	×	85.4%	71.2%	59.6%	49.7%	40.1%	3.06
	OpenVLA Kim et al. (2024)	✓	Discrete	✓	91.3%	77.8%	62.0%	52.1%	43.5%	3.27
	3DDA (Ke et al., 2024b)	×	Diffusion	×	93.8%	80.3%	66.2%	53.3%	41.2%	3.35
	VPP (Hu et al., 2024)	✓	Diffusion	×	95.7%	91.2%	86.3%	81.0%	75.0%	4.29
	Seer (Tian et al., 2024)	✓	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 (Chi et al., 2023)	×	Diffusion	×	86.3%	72.7%	60.1%	51.2%	41.7%	3.16±0.06
	RoboFlamingo (Li et al., 2023)	×	Cont.	✓	96.4%	89.6%	82.4%	74.0%	66.0%	4.09
	GR-1 (Wu et al., 2024)	✓	Cont.	×	94.9%	89.6%	84.4%	78.9%	73.1%	4.21
	MDT (Reuss et al., 2024b)	×	Diffusion	×	98.6%	95.8%	91.6%	86.2%	80.1%	4.52±0.02
	FLOWER (ours)	×	Flow	✓	**98.9%**	**96.7%**	**93.9%**	**90.2%**	**85.5%**	**4.62±0.03**
	FLOWER (ours)	✓	Flow	✓	**98.9%**	**99.2%**	**96.9%**	**92.3%**	**88.3%**	**4.67±0.04**

Table 1: **CALVIN Benchmark results for 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.

5 Evaluation

We aim to answer the following research questions through extensive evaluations: **(RQ I)** Does FLOWER deliver strong performance while significantly reducing computational demands compared to state-of-the-art VLA policies across diverse settings? **(RQ II)** Is FLOWER capable of learning a single policy that robustly handles robot embodiments (e. g., single-arm vs. dual-arm, delta-EEF vs. joint angles) and diverse action spaces? **(RQ III)** Does FLOWER generalize to unseen settings, novel objects and different lighting conditions? **(RQ IV)** How do the proposed design elements of our novel VLA architecture impact final performance?

To allow a fair comparison with the current state-of-the-art VLAs, we evaluate our model on the most commonly used simulation benchmarks for learning and generalization in IL: CALVIN (Mees et al., 2022), LIBERO (Liu et al., 2024a), and SIMPLER (Li et al., 2024b). In addition, we test FLOWER on simulated Bi-Manual Aloha environments (Zhao et al., 2023) to test its ability to deal with different action spaces and high frequency setups. Finally, we conduct a set of real-world experiments with a joint-state controlled Franka Robot to test its ability in generalization and cross-action space learning. Overall, FLOWER is evaluated across more than 190 tasks in 8 different benchmarks.

5.1 Language-Conditioned Multitask Learning

First, we evaluate FLOWER on the challenging generalization Benchmark CALVIN (Mees et al., 2022) to test its generalization and ability to scale with larger datasets.

Benchmark Description. CALVIN is a simulated benchmark designed to evaluate multitask tabletop manipulation. The benchmark consists of four distinct scene configurations (splits A-D) and features 34 fundamental tasks, supported by $24,000$ language-annotated demonstrations collected through human teleoperation equally distributed to each environment. Performance is measured by success rates on sequences of 1-5 consecutive tasks and the mean length of successfully completed task sequences (Avg. Len.). It has several challenges that test model scaling and generalization. We use the CALVIN ABC challenge, which is commonly used by many prior VLA (Wu et al., 2024; Tian et al., 2024; Li et al., 2024a; Hu et al., 2024) to demonstrate their generalization ability and the scaling benchmark CALVIN ABCD.

Baselines. We utilize current state-of-the-art VLA policies, that report results for the CALVIN benchmark as baselines. These include RoboFlamingo (Li et al., 2024a) and three generalist policies that rely on video prediction with inverse dynamics action generation: GR-1 (Wu et al., 2024), Seer (Tian et al., 2024), and *Video Prediction Policy* (VPP) (Hu et al., 2024). GR-1 learns to jointly predict future frames and actions after being pertrained on diverse video data. Seer uses continu-

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 Cross-X Pret	17	21	**8**	42	22
FLOWER Delta EEF Pret	**25**	**33**	0	**100**	**40**

Table 2: **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	30.4	32.7	2.0	35.8	25.2
FLOWER Delta EEF Pret	31.7	**46.3**	0.5	50.2	32.2

Table 3: **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.

ous video prediction like GR-1 but also conditions future actions on the latent tokens as an inverse dynamics approach, while VPP utilizes the latent representation of a finetuned Video Generation Model for conditioning a smaller Diffusion Transformer architecture for action generation. Additional baselines include MDT (Reuss et al., 2024a), a Diffusion Policy, that uses self-supervised learning objectives for reconstructing future frames and aligning goal-and image-conditioned state presentation with contrastive learning and a Diffusion Policy trained from scratch using the CNN backbone (Chi et al., 2023). The CALVIN benchmark has a standardized evaluation protocol to test models on a 1000 predefined instruction chain rollouts and report the average performance across these rollouts. This guarantees a fair comparison against prior reported results.

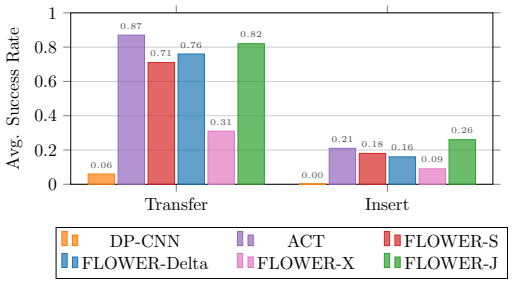
Results. The results in Table 1 show that FLOWER consistently outperforms all current state-of-the-art policies on both CALVIN challenges. Additionally, FLOWER requires only four denoising steps for action prediction, significantly fewer than the ten steps needed by Diffusion-based baselines like MDT. It sets a new record on both benchmarks while requiring very little compute for training. FLOWER outperforms all VLA policies with very low training time and high efficiency. On the generalist ABC challenge FLOWER achieves an almost perfect completion rate of 99.3% for the first task. On the scaling benchmark ABCD, FLOWER surpasses the prior state-of-the-art, MDT, by a considerable margin. These results not only confirm that FLOWER provides strong performance at low computational cost **(RQ I)**, but also demonstrate its ability to generalize to unseen environment settings better than previous VLA policies **(RQ III)**.

5.2 In-Context Tasks After Diverse Pretraining

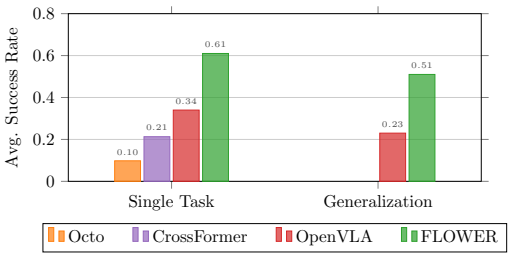
Next, we evaluate FLOWER on the SIMPLER benchmark (Li et al., 2024b) after pretraining on two dataset mixtures: a cross-embodiment mix and a delta-EEF only mixture (see Section 4.4 for details). SIMPLER is a real2sim benchmark that implements several scenes from the diverse BridgeV2 (Walke et al., 2023) and Google Robot setup (Brohan et al., 2023) to test foundation policies after pretraining. The benchmark requires policies to run approximately 3000 rollouts in different settings across two benchmarks with 8 different tasks in various conditions. A full overview for all tasks from SIMPLER is provided in Appendix F.

Baselines. On this setup, we compare FLOWER against RT-1X (Collaboration, 2023), Octo (Octo Model Team et al., 2023), OpenVLA (Kim et al., 2024), and CrossFormer (Doshi et al., 2024). 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 3 and the results for the Bridge challenge are shown in Table 2. Overall, FLOWER outperforms both Octo and OpenVLA on both benchmarks, despite having only 200 GPU hours of pre-training on heterogeneous robot



(a) ***Aloha Simulation Tasks:*** Average success rates on *Cube Transfer* and *Insertion*. *S* denotes trained from scratch, *Delta* uses delta-EEF pretraining, *X* applies cross-action space pretraining, and *J* represents droid joint-state pretraining.



(b) **Real World Results** for different Generalist Policies finetuned on a Franka Kitchen Setup. We evaluate policies on 20 different tasks involving pick and place and rigid-body manipulation. For generalization we evaluate novel objects, distraction scenes and different lightning conditions.

Figure 4: **Overview of simulation and real-world experiments. (Left)**: Evaluation of different pretraining strategies in Aloha simulation environment. **(Right)** Results from real-world experiments with a Franka Panda robot across 20 kitchen manipulation tasks.

datasets. Notably, the FLOWER variant pretrained on the delta-EEF mixture achieves stronger overall performance, with significant 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.

5.3 High Frequency Bi-Manual Control

Next, we evaluate FLOWER's ability to learn challenging high frequency control on the Aloha simulation setup (Zhao et al., 2023). We test several versions of FLOWER that have been finetuned on different pretraining mixes: Cross-X, Delta-EEF only 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) (Zhao et al., 2023). 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 3. The dataset contains 50 human-collected demonstrations for each tasks.

Results. As shown in Figure 4a, 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. Overall, the strong ability of FLOWER to adapt to new action spaces with strong results addresses our research question **(RQ II)**.

5.4 Real-World Setup

Next, we evaluate FLOWER in a real-world kitchen setting. Our evaluation covers 20 distinct tasks involving a variety of objects (e. g., pots, bananas, and toast). The demonstrations for training were collected by three human experts using kinesthetic teaching with a second Panda robot to gener-

ate long-horizon play demonstrations. To encourage diversity, the experts executed tasks in a randomized order, resulting in a rich dataset comprising 45 minutes of play data. We subsequently segmented and annotated this data into short-horizon sequences, each paired with natural language descriptions of the corresponding task, yielding a final dataset of 417 language-annotated trajectories. Policies are trained in joint state space at 6 Hz.

Baselines. We compare a finetuned version of FLOWER (pretrained on the cross-action space mix) against several common generalist and specialist policies, including Octo (Octo Model Team et al., 2023), OpenVLA (Kim et al., 2024), and CrossFormer (Doshi et al., 2024). For fair comparisons, we finetune each baseline on our dataset using our local cluster with recommended hyperparameters (details are provided in Appendix G).

Results. Each task is evaluated five times from randomized starting positions, resulting in a total of 100 evaluations per policy. The average performance per task is summarized in Figure 4b. FLOWER achieves the highest average success rate among all tested policies, doubling the performance of the second-best baseline, OpenVLA, (from 31% to 61%). These results underscore the robust performance of FLOWER across a diverse range of real-world kitchen tasks, highlighting its versatility in solving various tasks and thereby effectively addressing **(RQ II)**.

5.5 Real-World Generalization Capabilities

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%

Variant	Avg. Len.
FLOWER	**4.44±0.04**
- Flow Head	3.33±0.04
- Custom LR	0.8±0.04
- No VLM train	2.65±0.36
- Flow Transformer	3.67±0.06
+ Florence Decoder	4.40±0.03
+ small Florence	4.26±0.04
- VLM	3.42±0.07

Table 4: **Generalization and ablation evaluations. (Left)** Average success rates across different generalization scenarios for FLOWER and OpenVLA. **(Right)** Average Sequence Lengths for FLOWER Model Variants on the CALVIN ABC benchmark. Results are averaged over 1000 rollouts with 3 seeds each.

Finally, we test the generalization ability of FLOWER in our real-world setting across various distractions, different lighting conditions, and interactions with novel unseen objects. For these experiments, we use the best performing baseline from our initial real robot experiments, OpenVLA. Policies are evaluated on tasks under flashlight-only conditions, with additional tests incorporating various distracting objects scattered throughout the kitchen, as well as on tasks involving novel objects that were not encountered during training. As summarized in Figure 4b, FLOWER consistently outperforms OpenVLA across all tested scenarios, demonstrating superior adaptability and robustness. Detailed results for all experiments are summarized in Table 11 and Table 12. FLOWER surpasses OpenVLA by 28% on average across all generalization tests. Overall, these generalization experiments verify that FLOWER can effectively adapt to a wide range of unstructured, real-world variations by handling unseen scenarios and novel object interactions **(RQ III)**. However, FLOWER still struggles with some more complex challenges, e. g., "fine manipulation in highly cluttered environments", indicating areas for future improvement.

5.6 Ablations

Our ablation study to answer **(RQ IV)** on the CALVIN ABC benchmark (using 3 seeds, 1000 rollouts each) reveals several critical design elements: **VLM Backbone.** Replacing our VLM with a CLIP + ResNet-50 baseline significantly reduces performance (from 4.44 to 3.42), demonstrating the VLM's importance for encoding images and text. **Learning Rate Scheduling.** A dual learning rate strategy with separate schedulers for VLM and Flow Transformer components proves crucial. Without it, performance drops by 80% (Avg. Len. 0.8), likely due to the large value range of VLM output tokens destabilizing training. **VLM Fine-tuning.** Freezing the VLM reduces performance by

40% (Avg. Len. 2.65), highlighting the necessity of task-specific feature adaptation. **Architecture Choices.** Using Florence-2-large over base improves performance (4.44 vs 4.26), while removing the VLM decoder maintains comparable performance (4.42) while reducing parameters by 205M. A separate Flow Transformer outperforms using the VLM for velocity prediction (4.44 vs 3.67).

6 Conclusion

We presented FLOWER, an efficient Vision-Language-Flow policy that combines compact VLMs with rectified-flow-based action generation. Our Flow Transformer handles multiple control modalities within a 1B-parameter model. Across eight benchmarks spanning diverse simulation and real-world settings FLOWER achieves an average performance improvement of 32.7% with respect to the second best baseline in each setting. Overall, our work contributes a compact hybrid architecture that fuses semantic grounding from a pretrained vision-language model with expressive flow-based action generation, efficient low-compute pretraining, and cross-action space conditioning via intermediate VLM features. We will open-source both the model and training pipelines to support further research in efficient, generalist robotics.

References

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Nils Blank, Moritz Reuss, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Wenzel, Oier Mees, and Rudolf Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=EdVNB2kHv1.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Devin, Alex X Lee, Maria Bauza, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Max Braun, Noémie Jaquier, Leonel Rozo, and Tamim Asfour. Riemannian flow matching policy for robot motion learning. *arXiv preprint arXiv:2403.10672*, 2024.

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, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation. *arXiv preprint arXiv:2410.08001*, 2024.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, and Thomas Wolf. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch. https://github.com/huggingface/lerobot, 2024.

Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. Gr-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. Hardware conditioned policies for multi-robot transfer learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models. `https://arxiv.org/abs/2310.08864`, 2023.

Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, and Sergey Levine. The ingredients for robotic diffusion transformers. *arXiv preprint arXiv:2410.10088*, 2024.

Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *8th Annual Conference on Robot Learning*, 2024.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, 2024.

Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

Niklas Funk, Julen Urain, Joao Carvalho, Vignesh Prasad, Georgia Chalvatzaki, and Jan Peters. Actionflow: Equivariant, accurate, and efficient policies with spatially symmetric flow matching. *arXiv preprint arXiv:2409.04576*, 2024.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

Joey Hejna, Chethan Bhateja, Yichen Jian, Karl Pertsch, and Dorsa Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning. *arXiv preprint arXiv:2408.14037*, 2024.

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers. *arXiv preprint arXiv:2010.04245*, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations, 2024. URL `https://arxiv.org/abs/2412.14803`.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Xiaogang Jia, Denis Blessing, Xinkai Jiang, Moritz Reuss, Atalay Donat, Rudolf Lioutikov, and Gerhard Neumann. Towards diverse behaviors: A benchmark for imitation learning with human demonstrations. *arXiv preprint arXiv:2402.14606*, 2024.

Xinkai Jiang, Paul Mattes, Xiaogang Jia, Nicolas Schreiber, Gerhard Neumann, and Rudolf 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*, pp. 333–342, 2024.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024a.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *8th Annual Conference on Robot Learning*, 2024b. URL `https://openreview.net/forum?id=gqCQxObVz2`.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, et al. The bigscience roots corpus: A 1.6 tb composite multilingual dataset. *Advances in Neural Information Processing Systems*, 35:31809–31826, 2022.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024a.

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. *arXiv preprint arXiv:2405.05941*, 2024b.

Fanqi Lin, Yingdong Hu, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024a.

Peiqi Liu, Yaswanth Orru, Jay Vakil, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024b.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024c.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations. *arXiv preprint arXiv:2310.17596*, 2023.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

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, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. `https://octo-models.github.io`, 2023.

Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2): 1–179, 2018.

Austin Patel and Shuran Song. Get-zero: Graph embodiment transformer for zero-shot embodiment generalization. *arXiv preprint arXiv:2407.15002*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 3406–3413. IEEE, 2016.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal conditioned imitation learning using score-based diffusion policies. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning, 2024a.

Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. In *Robotics: Science and Systems*, 2024b.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From llms to actions: Latent codes as bridges in hierarchical robot control. *arXiv preprint arXiv:2405.04798*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021.

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

Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pp. 1723–1736. PMLR, 2023.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *arXiv preprint arXiv:2409.20537*, 2024.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu 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*, pp. 4818–4829, 2024.

Jonathan Heewon Yang, Dorsa Sadigh, and Chelsea Finn. Polybot: Training one policy across robots while embracing variability. In *Conference on Robot Learning*, pp. 2955–2974. PMLR, 2023.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Fan Zhang and Michael Gienger. Affordance-based robot manipulation with flow matching. *arXiv preprint arXiv:2409.01083*, 2024.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

	SIMPLER	CALVIN	LIBERO	Aloha	Real World Kitchen
Action Space Encoders			2-layered MLP		
Action Space Decoders			Linear Attention		
Number of Flow-T Layers			24		
Latent Dimension			1024		
Number of Heads			16		
Position Embedding			1D Rope		
Sampling Distribution			Logit Normal		
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	0,95B

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

A Extended Related Work

A.1 Cross-Embodiment Learning

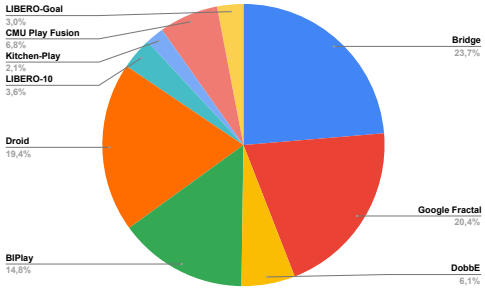
A core challenge in robotics is learning a unified policy for heterogeneous embodiments with distinct action and sensor spaces. Early approaches often applied modular policies (Huang et al., 2020) or hardware-conditioned representations (Chen et al., 2018), and some leveraged graph-based representations to generalize across different robot hands (Patel & Song, 2024; Yang et al., 2023). However, these efforts tended to focus on smaller datasets or simplified environments.

Recent work on large-scale cross-embodiment includes RoboCat (Bousmalis et al., 2023) and PolyBoT (Yang et al., 2023), which use action tokenization or hierarchical controllers, respectively. Liu et al. (2024c) propose a unified 258-dimensional action space (RDT-1B) with fixed action prediction length of 64 for all action spaces, while CrossFormer (Doshi et al., 2024) 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.

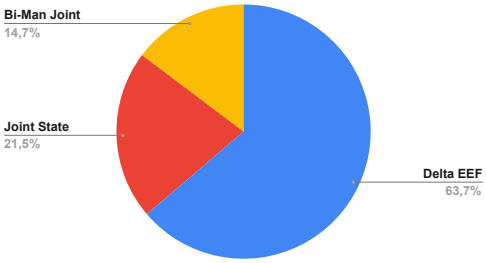
By incorporating an action-type Global AdaLN conditioned Flow Transformer with a pretrained VLM, FLOWER efficiently handles multiple embodiments while maintaining both action expressiveness and semantic understanding.

A.2 Rectified Flow and Diffusion Models in Robotics

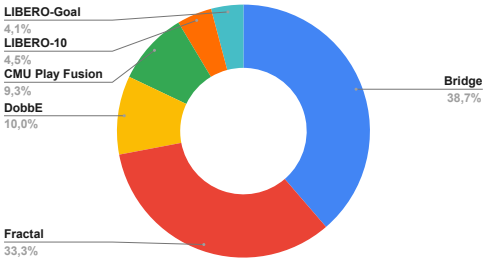
Diffusion models (Ho et al., 2020; Song et al., 2020; 2021) have become widely used for generating continuous robot actions from visual inputs (Chi et al., 2023; Octo Model Team et al., 2023; Reuss et al., 2023; Ke et al., 2024a), offering multi-modal behavior (Jia et al., 2024) and good scaling with large datasets (Octo Model Team et al., 2023; Liu et al., 2024c; Reuss et al., 2024a). More recently, *Rectified Flow* (Esser et al., 2024; Albergo & Vanden-Eijnden, 2022; Lipman et al., 2022)



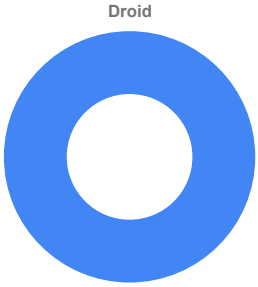
(a) **Overview of Cross-Action Space Dataset used for Pretraining FLOWER**



(b) **Action Space Distribution for the Cross-Action Space Pretraining Mix**



(c) **Delta-End-Effector Pretraining mix for FLOWER**



(d) **Joint-State only mix for Pretraining FLOWER**

Figure 5: **Overview of three pretraining mixes used for Prtraining FLOWER.**

has emerged as a promising alternative, enabling a straight-line probability path for action sampling that requires few discretization steps.

In robotic policy learning, ActionFlow (Funk et al., 2024) and others (Zhang & Gienger, 2024; Braun et al., 2024) showed rectified flow can generate actions more rapidly than standard diffusion. Such fast inference is crucial for high-frequency robot setups like Aloha (Fu et al., 2024). Yet, these works are typically confined to a single embodiment or a single action space. By contrast, FLOWER is the first open-source model to apply rectified flow as a *generalist* policy component, unifying expressive and multimodal flow-based action generation with diverse vision-language contexts.

B Pretraining Details

We pretrain FLOWER on different datasets mixes that are described in Figure 5a 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 (Collaboration, 2023) inspired by efforts from OpenVLA (Kim et al., 2024). 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.

C Pretraining Ablation Experiences

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

Hyperaprameter	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 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 7: **Hyperparmaeteres 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 (Doshi et al., 2024). 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 (Dasari et al., 2024) 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 images tokens and required GPU memory. Also given the more complex setting, we found 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 adopt 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 experiment 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.

D Details for Cross-Action Space Flow Transformer

Large transformers 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 (Shazeer, 2020), a sandwich-style MLP. that we express as follows. For an input vector $\mathbf{h}$,

$$\mathbf{y} \;=\; \Big(\mathrm{Norm}(\mathbf{W}_1\,\mathbf{h})\Big) \;\odot\; \mathrm{SiLU}\Big(\mathrm{Norm}(\mathbf{V}_1\,\mathbf{h})\Big), \tag{1}$$

where $\mathrm{SiLU}(x) = x\,\sigma(x)$, $\mathbf{V}, \mathbf{W}$ 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

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

yielding smoother gradients and reduced training instability (Zhang & Sennrich, 2019). Furthermore, we apply QK-value normalization (Henry et al., 2020) 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 (Esser et al., 2024) or Diffusion Policies (Liu et al., 2024c).

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.

E Benchmark Details

Benchmark	**FLOWER**	**2nd Best**	**Abs. Imp.**	**Rel. Imp. (%)**
CALVIN ABC	88.8%	85.8%	3.0%	3.5%
CALVIN ABCD	92.4%	90.4%	2.0%	2.2%
SIMPLER Bridge	40.0%	30.0%	10.0%	33.3%
SIMPLER Google	32.2%	42.4%	-10.2%	-24.1%
LIBERO	95.7%	76.5%	19.2%	25.1%
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	–	–	–	32.7%

Table 8: 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 relative improvement is computed over all benchmarks.

CALVIN Benchmark. (Mees et al., 2022) 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" → "lift blue block drawer" → "place in drawer" → "close drawer" → "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 exists three benchmark types: D→D, ABC→D, ABCD→D, that depend on the used dataset for trainign the policies. After training all are evaluated on the same environment D.

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

F SIMPLER BENCHMARK TASKS

The real2sim bencharmk SIMPLER (Li et al., 2024b) 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.

F.1 LIBERO BENCHMARK.

The LIBERO benchmark (Liu et al., 2024a) 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.

F.2 ALOHA BENCHMARK.

The Aloha benchmark (Zhao et al., 2023) 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(Zhao et al., 2023) and Diffusion Policy(Chi et al., 2023) implemented by lerobot (Cadene et al., 2024).

G REAL KITCHEN PLAY DATASET.

We conducted data collection through teleoperation, utilizing a leader-follower robot configuration to ensure precision and intuitive control (Jiang et al., 2024). 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.

H 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 40,0000 steps. Since FLOWER has been pretrained on single image, we extended the second static image for finetuning. No additinal modifications have been made to guarantee a fair comparsion 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.

	Spatial		Object		Goal		Long		90		Average (without 90)	
	SR (↑)	Rank (↓)	SR (↑)	Rank (↓)	SR (↑)	Rank (↓)	SR (↑)	Rank (↓)	SR (↑)	Rank (↓)	SR (↑)	Rank (↓)
Diff-P-CNN	$78.3 \pm 1.1\%$	4	$92.5 \pm 0.7\%$	2	$68.3 \pm 1.2\%$	4	$50.5 \pm 1.3\%$	5	-	-	$72.4 \pm 0.7\%$	4
Octo	$78.9 \pm 1.0\%$	3	$85.7 \pm 0.9\%$	4	$84.6 \pm 0.9\%$	2	$51.1 \pm 1.3\%$	4	-	-	$75.1 \pm 0.6\%$	3
OpenVLA	$84.7 \pm 0.9\%$	2	$88.4 \pm 0.8\%$	3	$79.2 \pm 1.0\%$	3	$53.7 \pm 1.3\%$	3	-	-	$76.5 \pm 0.6\%$	2
Baku	-	-	-	-	-	-	86.0	2	90.0	2	-	-
FLOWER	$\mathbf{97.1 \pm 2.1\%}$	1	$\mathbf{96.7 \pm 0.4\%}$	1	$\mathbf{95.6 \pm 0.6\%}$	1	$\mathbf{93.5 \pm 2.0\%}$	1	$\mathbf{93.2 \pm 1.2\%}$	1	$\mathbf{95.7 \pm 0.7\%}$	1

Table 9: **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 all tested settings.

H.1 Policy Adaption after Pretraining

Next, we evaluate FLOWER's ability to adapt to new robot environments after pretraining using the LIBERO benchmark suite (Liu et al., 2024a). LIBERO features a delta-EEF controlled Panda Robot operating in a variety of scenes and tests different aspects of policy learning, including spatial understanding, long-horizon task completion, and instruction following. We evaluate our pretrained model on joint-state libero dataset on all LIBERO variants: **Long**, **Spatial**, **Goal**, **Object**, and **90**. **Long** is the most challenging variant. It requires policies to solve ten long-horizon tasks involving several hundred steps. **90** assesses the ability to handle 90 diverse tasks across different scenes and objects. For each task, we perform 50 evaluations over three seeds to ensure a fair comparison (see Appendix F.1 for more details)

We compare FLOWER against both generalist policies such as OpenVLA (Kim et al., 2024) and Octo (Octo Model Team et al., 2023), as well as against the current state-of-the-art specialist policy Baku (Haldar et al., 2024), which uses a small transformer-based model with action chunking. As shown in Table 9, FLOWER significantly outperforms all baselines across every LIBERO variant, achieving near-perfect completion rates with success rates consistently above 93%. Notably, 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. These strong results complement our findings on CALVIN and SIMPLER, further demonstrating FLOWER's versatility and robustness across a range of robotic manipulation scenarios (**RQ I**).

H.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.

I 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 11 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 11. 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 6 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.

I.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.

J LIMITATIONS

Despite its advantages, FLOWER still has some limitations. FLOWER still requires iterative sampling procedure that is slower than a single forward pass from deterministic policies. So far, we have validated FLOWER primarily on three manipulation action spaces. Its ability to generalize to other embodiments, such as mobile navigation or humanoid locomotion, has not yet been explored. This remains an open area for future work. While FLOWER is considerably smaller than most state-of-the-art VLA models, its approximate 1B-parameter size may still present deployment challenges in low-resource settings. Finally, the pretraining performance for zero-shot deployment on the SIMPLER Google Robot benchmark indicates that further improvements are needed, requiring additional research.

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 10: **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**.

(a) Unseen Objects for Generalization Experiments

(b) Cluttered Scene with Distractors

Figure 6: **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.

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 11: **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 12: **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.

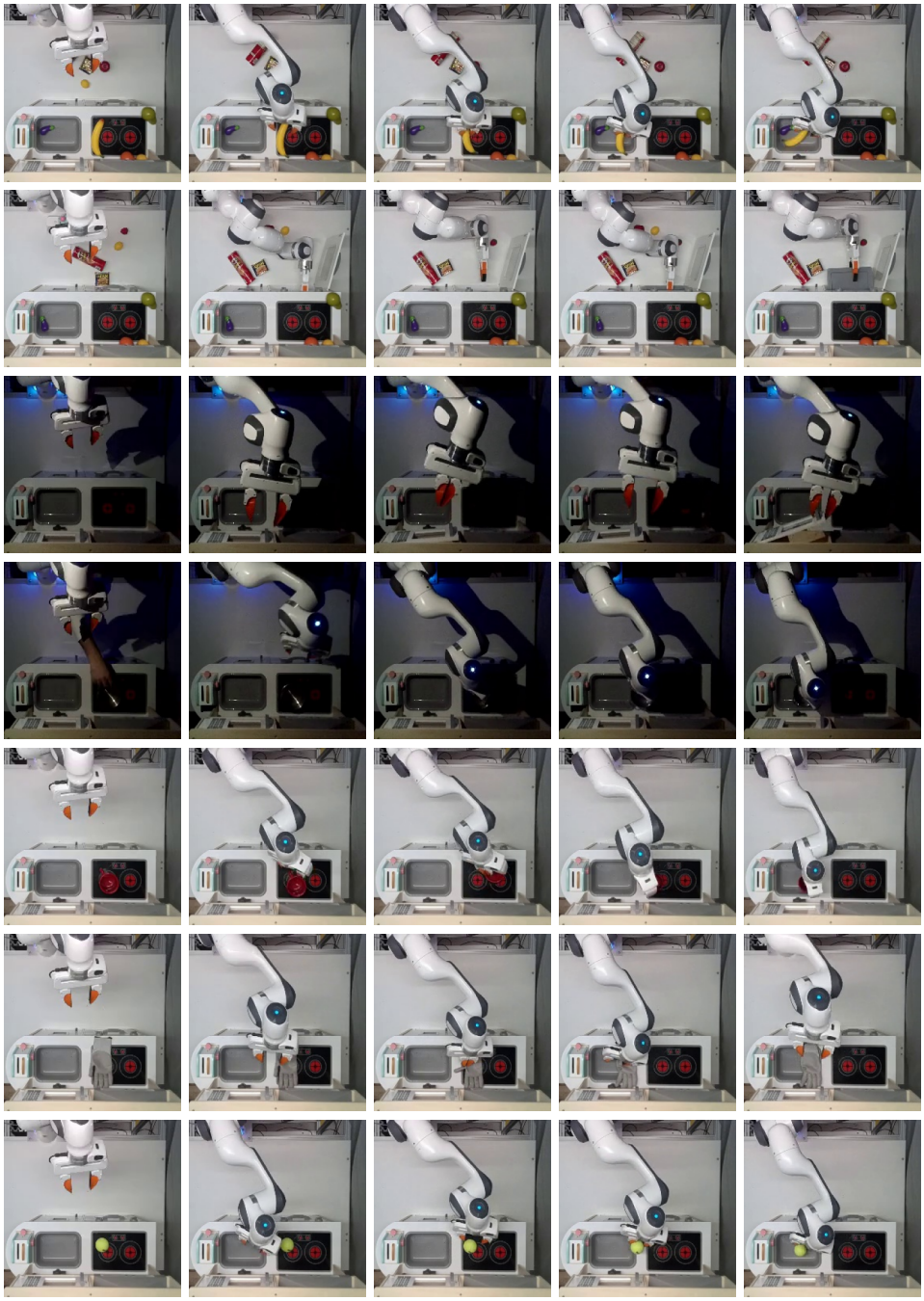


Figure 7: **Example generalization rollouts.** First two rows show rollouts with background distractors, rows 3 and 4 show rollouts with only a flashlight as a light source, and the last 3 rows showcase novel objects.