ContextFlow: In-Context Flow Matching for Robot Manipulation

Jian Ding[1]*, Xianjie Dai[1]*, Roei Herzig[2], Nussair Hroub[1], Jinjie Mai[1], Dengxin Dai[3], Bernard Ghanem[1], and Mohamed Elhoseiny[1]†

[1] KAUST, Thuwal, Saudi Arabia; {jian.ding, dai.xianjie, nussair.hroub, jinjie.mai, bernard.ghanem, mohamed.elhoseiny}@kaust.edu.sa
[2] University of California, Berkeley, USA; roeiherz@gmail.com
[3] Zurich, Switzerland; ddx2004@gmail.com

Abstract. Although highly effective in vision and language domains, applying in-context learning to robotics remains challenging. Existing autoregressive in-context imitation methods discretize continuous actions and exacerbate the accumulation of early prediction errors through next-token prediction, limiting their generalization on unseen task configurations. Meanwhile, flow-matching policies have been explored for continuous robot control and can help mitigate compounding errors; however, *in-context imitation learning within a flow-matching framework remains underexplored*. To address these limitations, we introduce ContextFlow, a *conditional flow-matching model* that learns continuous action distributions for in-context imitation learning. ContextFlow conditions flow-based action prediction on demonstrations and observations, enabling robust generation from noisy action distributions. To better encode multimodal in-context demonstrations, we adapt perceiver-style *multimodal context compressors* that distill visual, proprioceptive, and action sequences into compact, task-relevant latent representations. On LIBERO, ContextFlow outperforms ICRT by **35 percentage points** in average success rate on unseen task configurations, while matching the performance of the task-specific fine-tuned VLA model π_0 without any fine-tuning on unseen tasks. On real robots, it generalizes to unseen configurations of both single-arm and bimanual tasks, achieving **40%** success on a new pen-uncapping configuration. Project Page: https://dingjiansw101.github.io/contextflow-page/.

Keywords: Robot Manipulation · In-Context Learning · Flow Matching

1 Introduction

A hallmark of human intelligence is the ability to solve a new task after observing only a few demonstrations. In machine learning, this capability is often referred to as *in-context learning*: a model is conditioned at test time on a small set

* Equal contribution.
† Corresponding author.

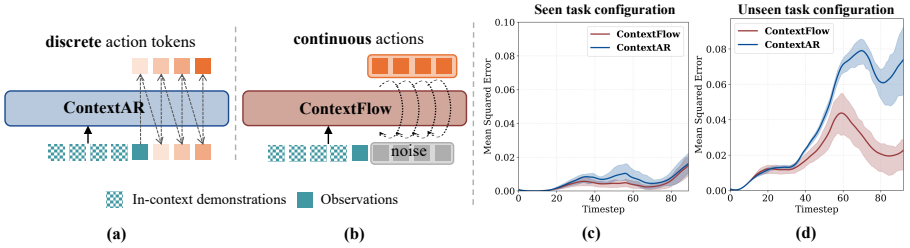

Fig. 1: Autoregressive (a) vs. Flow Matching (b) for In-Context Imitation Learning on action decoding and compounding errors. *In-context imitation learning* refers to performing a task conditioned on in-context demonstrations without updating model parameters. (a) In-context autoregressive models generate discretized actions step by step, which leads to compounding errors, especially under task distribution shift. (b) Our proposed in-context flow matching framework instead predicts a chunk of continuous actions, alleviating error accumulation and improving generalization. (c) Comparison of compounding errors under *seen* task configuration between autoregressive and flow-matching approaches. (d) Comparison of compounding errors under *unseen* task configuration.

of input-output examples and adapts to a new task without gradient updates. In-context learning has been prominently demonstrated in NLP: Large language models (LLMs) such as GPT-3 [7], trained with autoregressive objectives on massive text corpora, exhibit in-context learning as an emergent behavior [26,39].

Similar paradigms have been explored in robotics as *in-context imitation learning*, where demonstrations are represented as sequences of observations and expert actions. This promises to adapt a policy to a new task from only a few demonstrations, avoiding fine-tuning that existing imitation learning methods [27, 30, 31] require. Yet generalization in robotic manipulation is hard, as policies must produce continuous actions over long horizons from few demonstrations. Consequently, prior in-context imitation work [11, 32], like ours, targets *unseen configurations within related primitives* (*e.g.*, unseen objects or spatial arrangements) rather than arbitrary new tasks. Many existing in-context imitation learning approaches [11, 32, 38] instantiate the policy as an *autoregressive* model (ARM) [7, 36], a well-studied and scalable structure that can be directly adapted from LLMs. However, ARM-based policies (Fig. 1 (a)) face two fundamental limitations in robotic control. First, they predict actions sequentially, with each future token conditioned on previously predicted ones; as a result, *an early prediction error can propagate and compound over time*. Although this issue can be partially mitigated when training and evaluation are drawn from the *same* task configurations (see Fig. 1 (c)), it becomes much more pronounced in in-context imitation learning, where training and evaluation are drawn from *different* task configurations (see Fig. 1 (d)). Under such a distribution shift, early errors occur more frequently and are amplified throughout the autoregressive rollout. Second, many ARM-based policies rely on action tokenizers [28, 38]

that discretize continuous actions, introducing *reconstruction error* [28, 38] and further exacerbating execution failures.

In the broader imitation learning literature, action chunking [17, 45] or continuous action modeling, such as flow matching [4, 21, 22, 24], has been explored to alleviate reconstruction and compounding errors. Flow matching (see Fig. 1 (b)) operates directly in continuous action space and predicts an *action chunk in parallel* rather than *token by token*, which helps mitigate compounding errors, especially under distribution shift. However, unlike autoregressive in-context learning, whose architecture and training recipe are well-established in LLMs, *the incorporation of in-context learning into a flow-matching framework remains underexplored*. In this paper, we build on the architectural principles of a vision–language–action flow model, π_0 [4], and extend it to support multimodal in-context imitation learning. Concretely, we begin by building *ContextFlow-Plain*, which uses a *context expert* to encode the in-context demonstrations, along with the current observation, into a set of context tokens. An *action expert* then takes the current proprioceptive state and noisy action tokens, and refines them through a sequence of flow-matching updates while attending to the context tokens, ultimately predicting a clean *continuous* action chunk. By generating continuous action vectors *in parallel* and avoiding action discretization, ContextFlow-Plain mitigates the compounding errors that are especially harmful under unseen task configurations.

Although ContextFlow-Plain has already equipped a flow-matching framework with in-context learning ability, its performance is further constrained by the cost of encoding long, multimodal in-context demonstrations with a large vision–language–action encoder. Robot trajectories often span hundreds of timesteps and include camera observations, proprioception, and actions, all of which exhibit substantial spatial and temporal redundancy; processing these raw sequences directly with attention is prohibitively expensive. To address this efficiency bottleneck, we propose *ContextFlow*, which augments ContextFlow-Plain with *multimodal context compressors* adapted from perceiver-style architectures [14,15,19]. We tailor this structure to in-context demonstration encoding by distilling visual, proprioceptive, and action streams into a compact, fixed-size set of latent context tokens, preserving task-relevant information while substantially reducing the cost of context encoding and cross-attention.

We demonstrate the effectiveness of ContextFlow on the LIBERO simulation. To further evaluate ContextFlow on diverse and challenging real-world robot tasks, we collect a new dataset of expert demonstrations on the ALOHA platform. The dataset contains 1,318 multimodal trajectories across 25 task configurations, spanning both single-arm skills and challenging bimanual tasks that require high-frequency control and closed-loop feedback. Our contributions are three-fold:

- We formulate in-context imitation learning for robotics within a *conditional flow matching* framework, and empirically observe lower errors on unseen configurations.

- We curate a real-world ALOHA dataset for in-context imitation learning, featuring both single-arm tasks and fine-grained *bimanual* manipulation tasks.
- We evaluate ContextFlow both in LIBERO simulation and on ALOHA real robots. Across these settings, ContextFlow outperforms autoregressive in-context imitation learning baselines in our evaluations, while matching the performance of task-specific, fine-tuned large vision–language–action models without requiring any fine-tuning on unseen configurations.

2 Related Work

In-Context Imitation Learning. *In-context learning* enables adaptation from a few examples *without weight updates*, a capability popularized by autoregressive LLMs trained with next-token prediction [1,2,7,35]. In robotics, KAT [8] and RoboPrompt [42] perform few-shot imitation by tokenizing observations/actions and querying external LLMs, while earlier work [10] and Prompt-DT [41] rely on full-state observations. Instant Policy [37] and IMOP [44] use 3D structures, requiring external segmentation or IK execution; the latter can fail near singularities in fine manipulation. In contrast, ContextFlow uses only RGB and directly outputs executable actions. Similar to ours, ICRT [11] and LipVQ-VAE [38] train end-to-end on RGB observations using autoregressive action tokens, with LipVQ-VAE improving tokenization, and RICL [34] extends autoregressive VLA models with in-context adaptability. We instead formulate in-context imitation learning via conditional flow matching, modeling continuous action chunks directly without action discretization. We further demonstrate real-world *fine-grained bimanual* manipulation requiring *high-frequency* control and *closed-loop* feedback, a setting still underexplored in prior work.

Vision Language Action Models. Vision language action (VLA) models extend vision language models (VLMs) to predict low-level robotic actions from visual observations and language instructions. RT-1 [6], RT-2 [5], and OpenVLA [18] adopt autoregressive architectures trained with a next-token prediction objective over discrete action tokens. To improve token efficiency and reduce discretization artifacts, Fast Tokenizer [28] proposes more compact action vocabularies, and OpenVLA-OFT [17] shows that adding a parallel continuous-action prediction head can further boost performance. More recent VLA models [4, 25, 29, 40] incorporate diffusion [20] or flow-matching [22] heads for continuous control, with π_0 [4, 29] demonstrating strong instruction-following capabilities. However, these models primarily rely on textual instructions, which limit their generalization ability to new task configurations. In contrast, our approach uses *multimodal in-context demonstrations* as the task prompt. VIMA [16] also studies multimodal in-context prompting for robotics, but predicts only *2–5 high-level* actions (e.g., "pick and place" or "wipe") per episode, whereas our model predicts hundreds of low-level action sequences.

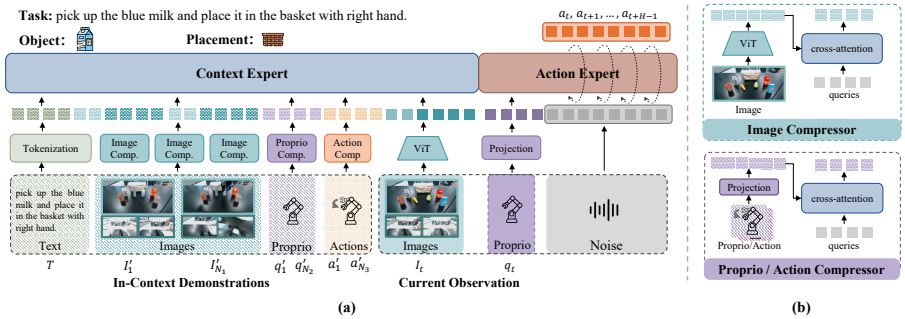

Fig. 2: ContextFlow Model Architecture. We follow the mixture-of-experts design and employ two experts: the **context expert** and the **action expert**. The current observation, text prompt, and in-context demonstrations are first encoded and passed through *multimodal context compressors*, and the resulting compressed tokens are routed to the context expert, while the current proprioceptive state and action noise tokens are routed to the action expert. The context expert encodes the current observation and contextual information, while the action expert performs flow matching steps to predict the actions.

3 Method

3.1 Problem Formulation

Learning Objective. We assume a distribution of tasks $\tau \sim \mathcal{T}$, where each task τ corresponds to a specific goal the robot must achieve, described by a textual instruction T (e.g., *"put the bowl on top of the cabinet"*). For each task τ, demonstrations are drawn from $\mathcal{D}(d \mid \tau)$, where each demonstration is represented as $d = [T, X, Q, A]$: a task instruction T, an image sequence $X = (I_1^{'}, \ldots, I_{T_d}^{'})$, proprio states (e.g., joint angles, gripper states) $Q = (q_1^{'}, \ldots, q_{T_d}^{'})$, and actions $A = (a_1^{'}, \ldots, a_{T_d}^{'})$. Each image $I_t^{'} \in \mathbb{R}^{H \times W \times 3}$, proprio state $q_t^{'} \in \mathbb{R}^{D_s}$, and action $a_t^{'} \in \mathbb{R}^{D_a}$ are recorded at step t of a trajectory of length T_d. Our goal is to model the target data distribution: $\pi(\mathbf{a}_t \mid o_t, d)$, where we define $\mathbf{a}_t = a_{t:t+H-1}$ as the chunk of H future actions. $o_t = (I_t, q_t)$ is the current observation and d is the provided in-context demonstration. I_t is the current image and q_t is the current proprio state. $\pi(\mathbf{a}_t \mid o_t, d)$ can be learned through any proper *conditional generative model*.

Flow Matching Conditioned on Demonstration. While prior work [38] used an *autoregressive model* to learn the distribution $\pi(\mathbf{a}_t \mid o_t, d)$, it requires tokenizing continuous actions into discrete tokens, which introduces quantization error and reduces accuracy. To effectively model the conditional distribution $\pi(\mathbf{a}_t \mid o_t, d)$, we leverage the Flow Matching (FM) framework, which learns a deterministic, time-dependent vector field that smoothly transports samples from a simple Gaussian prior to the target conditional distribution, allowing for direct modeling in *continuous action spaces* without discretization. Given the

current observation o_t and the provided demonstration d, we construct a family of intermediate distributions that smoothly interpolate between a standard Gaussian prior at $s = 0$ and the target distribution $\pi(\mathbf{a}_t \mid o_t, d)$ at $s = 1$:

$$p_s(\mathbf{a}_t^s \mid \mathbf{a}_t) = \mathcal{N}(\mathbf{a}_t^s \mid s\,\mathbf{a}_t, (1-s)^2 I), \quad s \in [0,1]. \tag{1}$$

To sample intermediate actions, we draw noise $\epsilon \sim \mathcal{N}(0, I)$ and linearly interpolate: $\mathbf{a}_t^s = (1-s)\,\epsilon + s\,\mathbf{a}_t$. For this linear interpolation path, the corresponding conditional velocity field can be computed in closed form [22]: $u(\mathbf{a}_t^s \mid \mathbf{a}_t) = \mathbf{a}_t - \epsilon$, representing the direct displacement from the noisy action chunk to the clean one. We parameterize our learnable vector field with a neural network: $u_\theta(\mathbf{a}_t^s, o_t, d, s)$, which conditions on the current noisy action chunk, observation, demonstration, and interpolation time s (encoded with sinusoidal embeddings).

We train our model by minimizing the following Conditional Flow Matching loss:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{s,\,\mathbf{a}_t,\,\epsilon} \left\| u_\theta(\mathbf{a}_t^s, o_t, d, s) - u(\mathbf{a}_t^s \mid \mathbf{a}_t) \right\|^2, \tag{2}$$

where $s \sim \mathrm{Beta}(\alpha, \beta)$, $\mathbf{a}_t \sim \pi(\mathbf{a}_t \mid o_t, d)$, $\epsilon \sim \mathcal{N}(0, I)$.

During inference, we generate action sequences by integrating the learned velocity field from 0 to 1:

$$\mathbf{a}_t^{s+\delta} = \mathbf{a}_t^s + \delta\, u_\theta(\mathbf{a}_t^s, o_t, d, s), \quad \mathbf{a}_t^0 \sim \mathcal{N}(0, I), \tag{3}$$

using a simple forward Euler solver with step size $\delta = \frac{1}{K}$ ($K = 10$ steps in our implementation). The resulting action chunk $\mathbf{a}_t^1$ provides the policy's next predicted actions.

3.2 Model Architecture

Overview. To build the conditional flow matching model, we design the framework illustrated in Fig. 2. We introduce a *context expert* and *multimodal context compressors* to efficiently encode in-context demonstrations. Following the mixture-of-experts design [4, 9, 33], the context expert encodes observation and compressed demonstration tokens, while the action expert processes proprio and action noise tokens, which interact with context tokens through attention. The action expert predicts a clean action chunk of horizon H through K flow-matching steps.

Multimodal Context Compressor. Encoding long multimodal demonstrations is challenging due to the *quadratic complexity of attention* over long sequences. Even tasks like "put egg in box" may produce demonstrations of length $T_d \approx 800$ steps, and multiple camera views introduce additional high-dimensional visual inputs $I_t^{'} \in \mathbb{R}^{H \times W \times 3}$ that further increase computational cost. To reduce computational cost, we first perform temporal subsampling for each modality, resulting in shorter sequences of lengths N_1, N_2, and N_3 for images, states, and actions, respectively. To further reduce the redundancy, we adopt perceiver-style [14, 15] *learnable context compressors* that condense multimodal demonstrations into compact latent representations while preserving task-relevant information. As shown in Fig. 2, the learnable context compressors then take the

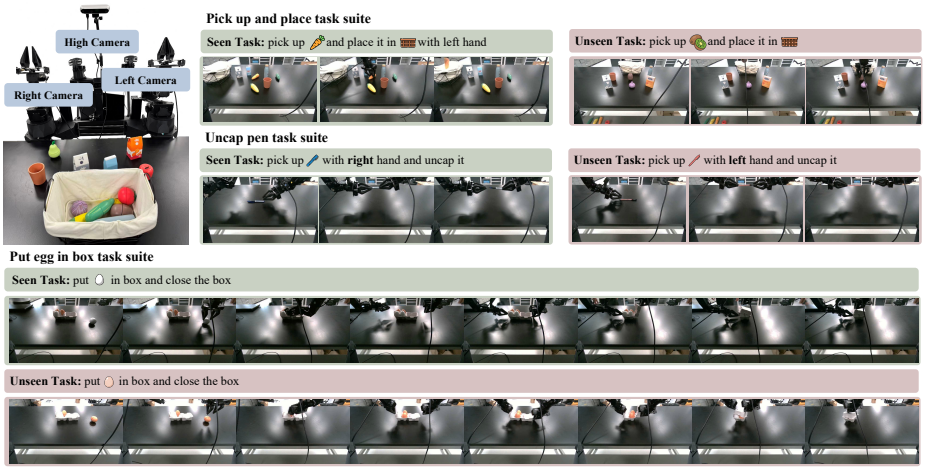

Fig. 3: Examples of our real-world in-context experimental setup using the ALOHA robot platform. We evaluate three task suites. Each suite comprises multiple task configurations that differ in manipulated objects or the required left–right hand motion.

input token sequences $\mathbf{X} \in \mathbb{R}^{S \times D}$ from each modality and map them into fixed-size latent representations using alternating cross-attention and self-attention layers, where S denotes the input sequence length and D the embedding dimension. A set of N_q learnable query tokens $\mathbf{Z}_0 \in \mathbb{R}^{N_q \times D}$ forms the initial latent representation. The compressor then refines these latents through L layers that alternate between cross-attention and self-attention, each sub-layer using pre-normalization and a residual connection. The cross-attention layers let the latents $\mathbf{Z}$ attend to the input tokens $\mathbf{X}$ to absorb information from the demonstration, while the self-attention layers let the latents attend to one another and are followed by a token-wise feed-forward network. After the final layer, the compressed representation is the latent set $\mathbf{Z}_{\text{out}} \in \mathbb{R}^{N_q \times D}$.

We design three separate compressors, $\mathcal{C}_{\text{img}}$, $\mathcal{C}_{\text{state}}$, and $\mathcal{C}_{\text{action}}$, to encode image, state, and action sequences respectively into compact latent sets of equal length. The image compressor $\mathcal{C}_{\text{img}}$ is applied to each image and shared across camera views and time steps, producing per-view latent embeddings that capture spatial information before temporal aggregation across frames. The outputs of all compressors are concatenated to form a unified context representation:

$$\mathbf{T}_{\text{context}} = [\mathbf{Z}_{\text{img}}; \mathbf{Z}_{\text{state}}; \mathbf{Z}_{\text{action}}] \in \mathbb{R}^{N_{\text{context}} \times D}, \tag{4}$$

where N_{context} is the total number of latent tokens after concatenation.

The task instruction is tokenized using a text tokenizer [3], and the current observation images are encoded with the same ViT used for demonstration frames. These tokens, together with the compressed demonstration context, are concatenated and passed to the context expert. During training, the context expert receives tokens from the current observation of a task (e.g., task 1) along

with randomly sampled demonstrations from the same task. At inference time, demonstrations from a novel task (e.g., task 2) are supplied as in-context examples, allowing the model to perform previously unseen task configurations by leveraging the demonstration-driven context without any fine-tuning.

Context-Conditioned Flow Matching. The current proprio state of the robot arm and the noise action tokens are routed to the action expert, where they attend to the context expert tokens. This attention mechanism allows the action expert to aggregate relevant information from both the in-context demonstration and the current observation, enabling action prediction on unseen task configurations. The noise action tokens are then updated through K flow-matching steps, each involving attention interactions between the noise action tokens and the context expert tokens. To enhance the efficiency of flow matching, we implement a block-wise causal attention mechanism that enforces a causal dependency from context to action tokens: action tokens can attend to context tokens, but context tokens cannot attend to action tokens. This causal design allows the caching of context tokens during inference, eliminating the need to recompute context tokens at each flow matching step.

4 Experimental Setup

4.1 Simulation Benchmark Setup

Simulation Benchmark. We compare ContextFlow with the baselines on the LIBERO benchmark [23], a simulated environment featuring a Franka Emika Panda robotic arm. The benchmark includes multimodal demonstrations consisting of *RGB images* from the workspace and wrist cameras, *robot proprioception*, and *text task instructions*. Our evaluation focuses on two task suites: spatial reasoning (LIBERO-Spatial) and object manipulation (LIBERO-Object). To increase the diversity of training data, we additionally include demonstrations from LIBERO-Goal and LIBERO-10 during training.

Training and Testing Split. Each suite comprises 10 unique tasks with a total of 500 expert demonstrations per suite. We divide the 10 tasks of each evaluated task suite into 8 *seen tasks* and 2 *unseen task configurations*. This split is randomly predefined in the benchmark setup and consistent across all experiments. Seen tasks are used during training, while unseen task configurations are reserved for inference. We focus our evaluation on *unseen configurations within related primitives*, which reflect the model's generalization ability. We report the average success rate over 50 inference trials per task.

4.2 Real-World Experiment Setup

We conducted real-world experiments on an ALOHA robot platform, as shown in Fig. 3. The setup includes one high camera and two wrist cameras, and demonstrations were collected through human teleoperation. We evaluated on three task suites: (1) "pick up and place," following the instruction template `‘‘Pick up`

`the <object> and place it in the basket with <left/right> hand.''` (2) "uncap pen," following the instruction template `‘‘Pick up <type of pen> with <left/right>, grasp the cap with another hand and uncap it.''` (3) "put egg in box," following the instruction template `‘‘Pick up <object> with right hand and place it in box and close the box.''` For each task suite, there are different *task configurations* with different *objects* or *left–right hand motion trajectories*. In total, these three task suites comprise 21 task configurations and 1,064 training demonstrations. The average episode lengths are 234, 534, and 736 timesteps, respectively. Notably, the latter two suites involve *long-horizon*, fine-grained bimanual manipulation requiring precise closed-loop feedback. We also included 4 extra bimanual tasks (handover, cup stack, stir, and water wipe) with 254 demonstrations for training only. Examples of seen and unseen task configurations are shown in Fig. 3. Detailed task names and stats are provided in the *Supplementary Material*.

4.3 Implementation Details

We use SigLIP [43] as the vision encoder, processing observation images at a resolution of 224×224. Observation images at each timestep consist of all camera views. The transformer architectures of the context expert and the action expert follow a structure similar to Gemma [12], but are significantly smaller. We use a 300M model for the context expert. By default, the model weights for both SigLIP and the action expert are initialized from π_0 [4]. The weights of the context expert are randomly initialized. We use LoRA [13] to fine-tune the action expert, with a rank of 32 and an α of 32. The vision encoder and context expert was fully fine-tuned. We train the models for 20K iterations with a batch size of 32. We use a cosine learning rate schedule with a peak of 2.5×10^{-5} and a minimum of 2.5×10^{-6}, including a warm-up phase over the first 1,000 steps. For in-context demonstrations, we sample $N_1 = 8$ image frames and $N_2 = 128$ proprioceptive states and $N_3 = 128$ actions. Each compressor uses 32 learnable query tokens per modality and is refined through stacked cross-attention and self-attention layers: the image compressor has 4 layers, and the proprioceptive state and action compressors each have 2 layers. For flow matching, we set the Beta distribution parameters to $\alpha = 1.5$ and $\beta = 1$.

4.4 Baselines

Autoregressive-Based In-context Imitation Learning Methods. We compare with autoregressive baselines to show the advantage of flow matching in the in-context imitation learning task setting. We compare with the ICRT [11], which is a recently proposed in-context imitation learning model that formulates the learning process as a next-token prediction task. Images and proprioceptive states are encoded using attention pooling to obtain compact state representations. Demonstrations are structured as sequences of states and actions. A causal Transformer (*i.e.*, LLaMA2 [36]) takes as input the current state, the history of past states and actions, and demonstration sequences to predict the

Table 1: Comparison of models on LIBERO simulation. Models are trained on seen tasks and evaluated on unseen task configurations. The two unseen task configurations on LIBERO-Spatial are "pick up the black bowl on the cookie box and place it on the plate" and "pick up the black bowl next to the plate and place it on the plate." The two unseen task configurations on LIBERO-Object task suite are "pick up the milk and place it in the basket" and "pick up the tomato sauce and place it in the basket." FT means fine-tuning on unseen task configurations with 1 demonstration. ContextAR is our improved in-context autoregressive baseline for fair comparison with ContextFlow.

Model	LIBERO-Spatial			LIBERO-Object			Avg.
	$\mathbf{S}_1$ (bowl on box)	$\mathbf{S}_2$ (bowl next to plate)	**Avg.**	$\mathbf{O}_1$ (milk in basket)	$\mathbf{O}_2$ (sauce in basket)	**Avg.**	
OpenVLA-OFT [17]	**94.0**	32.0	63.0	0.0	34.0	17.0	40.0
π_0 [4]	52.0	0.0	26.0	46.0	80.0	63.0	44.5
π_0 [4] FT	94.0	32.0	63.0	80.0	84.0	82.0	72.5
ICRT (AR baseline) [11]	54.0	0.0	27.0	4.0	**96.0**	50.0	38.5
ContextAR (AR baseline$^+$)	**94.0**	12.0	53.0	16.0	92.0	54.0	53.5
ContextFlow-Plain (Ours)	90.0	20.0	55.0	**92.0**	54.0	73.0	64.0
ContextFlow (Ours)	86.0	**42.0**	**64.0**	76.0	90.0	**83.0**	**73.5**

next action within a specified horizon. Following the original ICRT implementation, we use LLaMA2-Base as the backbone LLM. During inference, an episode from the corresponding task will be provided as the prompt. ICRT uses modalities of images, and state/actions as in-context demonstration. In addition to ICRT, we implemented an autoregressive baseline, the *in-context autoregressive model (ContextAR)*, which uses the same vision backbone as ContextFlow and is initialized from π_0 to ensure a fair comparison between autoregressive and flow-matching approaches. ContextAR accepts all modalities: images, text, and states/actions as context.

Large Scale Vision-Language-Action (VLA) Models. These VLA models are pre-trained on large-scale robot data and fine-tuned on seen tasks of LIBERO data. We provide text instructions of unseen task configurations to these VLA models to prompt them to predict actions for unseen task configurations. We compared with π_0 and OpenVLA-OFT [17].

VLA Fine-tuned with 1-shot Demonstration from Unseen Task. VLA models cannot accept in-context demonstrations as input. Therefore, we further fine-tuned π_0 with one demonstration from an unseen task. Although task-specific fine-tuning methods are not as flexible as in-context learning methods that do not require fine-tuning, they can serve as a reference.

5 Results

5.1 Comparison with Baselines

Autoregressive vs. Flow-Matching Models. As shown in Tab. 1, we compare flow-matching policies against autoregressive (AR) baselines in a controlled manner. We treat ICRT [11] as a representative autoregressive baseline, and

introduce ContextAR as a stronger autoregressive baseline that incorporates π_0 [4, 28] pre-training. To isolate the effect of the modeling paradigm (autoregression vs. flow matching) from other architectural choices, our key comparison is between ContextFlow-Plain (flow matching, without context compressors) and ContextAR. Both models use the same vision backbone and are initialized from π_0, making this our most controlled comparison between autoregressive and flow-matching action modeling. Under this setup, ContextFlow-Plain outperforms ContextAR by 10.5 percentage points in success rate, providing evidence that flow matching is beneficial for in-context imitation learning. ContextFlow also achieves higher throughput than AR baselines, as shown in the Supplementary Material.

Text vs. In-Context Demonstrations. We compared in-context imitation learning models with VLA models. The results show that in-context imitation learning models are usually better than VLA models. ContextAR outperforms OpenVLA-OFT [17] by 13.5 percentage points (see Tab. 1), showing that a text prompt alone is insufficient for the generalization to unseen tasks. Providing detailed in-context demonstration (sequence of images, states, and actions) benefits the generalization to unseen task configurations.

Parametric Fine-tuning vs. In-Context Learning. In our in-context learning setting, methods receive a single demonstration from each unseen task at test time. To compare against parametric adaptation, we additionally fine-tune the VLA model π_0 with one demonstration from each unseen task at test time, denoted π_0 (FT). Our proposed ContextFlow performs comparably to π_0 (FT) in success rate (73.5% vs. 72.5%; see Tab. 1). Note that the in-context learning paradigm does not require any fine-tuning on demonstrations from unseen task configurations, enabling much faster adaptation. Moreover, aside from the additional computation cost, task-specific fine-tuning can also lead to catastrophic forgetting of previously learned tasks. These results highlight the advantage of our in-context learning model over baselines with task-specific fine-tuning.

5.2 Ablations

The Effectiveness of Context Compressor. As shown in Tab. 1, adding context compressors further boosts performance, with ContextFlow outperforming ContextFlow-Plain by 9.5 and ContextAR by 20 percentage points in average success rate. We conduct an ablation to examine the contribution of the proposed context compressors for different modalities in the in-context demonstrations. As shown in Tab. 2, enabling only the image compressor improves the average success rate by 4 percentage points. Activating both compressors improves overall performance over ContextFlow-Plain, with the largest gains in spatial reasoning and average success rate, indicating that jointly compressing visual and state/action contexts is important for effective in-context flow matching.

Influence of Context Modalities. The text prompting and in-context demonstrations include three modalities: (1) text instruction, (2) images, and (3) proprioceptive states and actions. To assess the contribution of each modality to in-context imitation learning, we conducted ablation experiments by removing

one modality at a time from our full model, as shown in Table 3. The results show that all three modalities are important for both seen and unseen task configurations, with a greater impact observed on unseen performance. Removing the text modality leads to a 9.0 points drop in performance on average for unseen task configurations. Among all modalities, proprioceptive states and actions are the most critical: removing them causes a 44.5 points drop on average for unseen task configurations.

Hyperparameter Analysis. We further study how the length of in-context demonstrations influences model performance in Tab. 4. When we change the number of frames from 4 to 8 while keeping the number of states/actions the same, there is an improvement by 3.0 points in success rate on average, suggesting more visual information would help in-context imitation learning. We then fix the number of frames, and vary the number of states/actions, and find that 128 is the optimal number for states/actions in context.

Comparison with Other Action Heads. To further study the design choices within continuous parallel action heads, we compare MLP L1 regression, diffusion, and flow matching. The success rates on LIBERO are reported in Tab. 5. MLP L1 regression achieves a success rate of 58%, outperforming ContextAR by 4.5 points, which suggests that continuous parallel prediction is effective even with a simple regression head. The diffusion head converges slowly in our setting, achieving 16.5% success after 20k iterations and 33.5% after 100k iterations. Among the continuous parallel action heads, flow matching achieves the best overall performance with a success rate of 73.5%. We therefore adopt flow matching as the default action head, which is also aligned with the π_0 pretraining objective.

Results on Expanded Training Data and Evaluation Tasks. In the previous models, we demonstrated generalization only on LIBERO-Spatial and LIBERO-Object. We investigate whether generalization on these more challenging task suites can be improved by increasing the amount of training data. To this end, we incorporate LIBERO-90 into ContextFlow's training set and evaluate on additional unseen tasks from LIBERO-Goal and LIBERO-10, with two unseen tasks from each suite. The results are reported in Tab. 7. With this more diverse training data, ContextFlow achieves improved results on the previously unsolved LIBERO-Goal and LIBERO-10 tasks, raising the overall average from 36.8 to 43.0 points.

5.3 Results on Real-World Experiments

We compared the proposed ContextFlow with ICRT [11] in real-world experiments in Tab. 8. Each model was evaluated for 10 trials per task. ContextFlow achieved reasonable performance on both single-arm pick-and-place configurations and fine-grained bimanual configurations. In contrast, ICRT succeeded in only 2/10 trials on the pear pick-and-place task and struggled with the fine-grained bimanual tasks, which require high-frequency control and closed-loop feedback. These results suggest that ContextFlow can reliably leverage

Table 2: Ablation study on context compressors.

Model	$\mathcal{C}_{\text{img}}$	$\mathcal{C}_{\text{state}}$ & $\mathcal{C}_{\text{action}}$	Spatial	Object	Avg.
ContextFlow-Plain	✗	✗	55.0	73.0	64.0
	✓	✗	51.0	**85.0**	68.0
ContextFlow	✓	✓	**64.0**	83.0	**73.5**

Table 3: Ablation study on different modalities in the demonstrations. * indicates the default model.

	State	Image	Text	Spatial	Object	Avg.
	✗	✓	✓	30.0	28.0	29.0
	✓	✗	✓	42.0	48.0	45.0
	✓	✓	✗	59.0	70.0	64.5
*	✓	✓	✓	**64.0**	**83.0**	**73.5**

Table 4: Ablation study on the number of images and states in in-context demonstrations.

	# Images	# States	Spatial	Object	Avg.
	4	128	62.0	79.0	70.5
	8	64	53.0	**87.0**	70.0
*	8	128	**64.0**	83.0	**73.5**
	8	256	52.0	81.0	66.5

Table 5: Ablation of action heads.

Action Head	Iterations	Spatial	Object	Avg.
MLP L1	20k	26.0	**90.0**	58.0
Diffusion	100k	30.0	37.0	33.5
Flow Matching	20k	**64.0**	83.0	**73.5**

Table 6: Cumulative MSE.

Model	MSE (m^2)
ContextAR (baseline$^+$)	4.7
ContextFlow-Plain	3.0
ContextFlow	2.2

Table 7: Results with Expanded Training Data and Evaluation Tasks

Training Data	Spatial	Object	Goal	10	Avg
4 suites	**64.0**	83.0	0.0	0.0	36.8
4 suites + LIBERO-90	55.0	**85.0**	**29.0**	**3.0**	**43.0**

in-context demonstrations to solve unseen real-world manipulation task configurations. More real-world experiments and rollout videos are provided in the *Supplementary Materials*.

6 Analysis

Compounding Error. To quantify compounding error, we compute the mean squared error (MSE) between the evaluation rollout and the reference expert trajectory over time. We visualize the error on a two-stage pick-and-place task (S_2 in Tab. 1) using 50 test episodes, plotting the mean and the 20–80 percentile band in Fig. 1 (d). We observe a mild decrease in MSE starting around step 60, aligning with the transition that marks the end of stage 1 (approximately steps 50–65). ContextFlow and ContextAR exhibit similar errors during the first 30 steps; however, beyond this point, ContextAR's error grows noticeably over time. In this evaluated task, this suggests that autoregressive models tend to accumulate larger compounding errors than flow-matching models in robot manipulation. The distribution shift between seen and unseen configurations further

Table 8: Performance on unseen task configurations on a real-world robot. See Sec. 4.2 for task instruction templates. Elements in different task configurations are listed in the table.

Model	Single-arm Pick-and-place tasks				Fine-grained bimanual tasks	
	Left hand		Right hand		Uncap (Left Hand)	Put in box
	Pear	Orange	Banana	Kiwi	Red Pen	Red Egg
ICRT [11]	**2/10**	0/10	0/10	0/10	0/10	0/10
ContextFlow	**2/10**	**5/10**	**4/10**	**4/10**	**4/10**	**6/10**

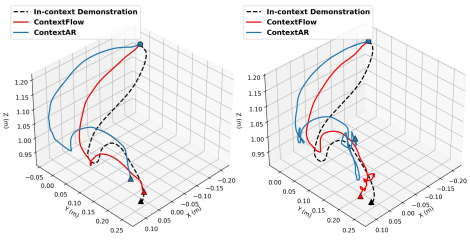

Fig. 4: Visualization of in-context demonstration, and success, failure of ContextFlow and ContextAR. Left: Success trajectories. Right: Failure trajectories.

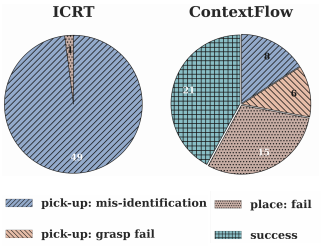

Fig. 5: Failure mode analysis. We analyze rollouts of ICRT and ContextFlow and categorize failure modes into three categories.

exacerbates this compounding effect and leads to lower success rates on unseen configurations. We also visualize the expert in-context demonstration trajectory and the rollout end-effector trajectories in 3D space. Further, we sum the per-step errors over the trajectory as a metric. From Tab. 6, we can see that both flow matching reduces MSE accumulation by 1.7, and compression can further reduce MSE by 0.8. Successful rollout trajectories are shown in Fig. 4 left. Although both ContextFlow and ContextAR succeed, the ContextFlow trajectory aligns more closely with the expert demonstration. Failed trajectories are shown in Fig. 4 right, where we observe clear detours in the pick-up and place stages, respectively. See more detailed analyses on compounding error in *Supplementary Materials*.

Failure Mode. We group failure modes into three categories at different stages: *pick-up: misidentification*, *pick-up: grasp fail*, and *place: fail*. As shown in Fig. 5, ICRT frequently fails in the first stage by reaching a location near, but not on, the target object. This may stem from its state-token design: the visual latent and proprioceptive latent are attention-pooled into a single token, potentially discarding spatial detail needed for accurate object identification. For ContextFlow, the failure cases are more evenly distributed across the three stages with more examples on the last one. We provide visualization of different failure modes and analysis in *Supplementary Materials*.

Attention in Image Compressor. To understand what the image compressor learns, we visualize the attention of its learnable queries over frames in the in-

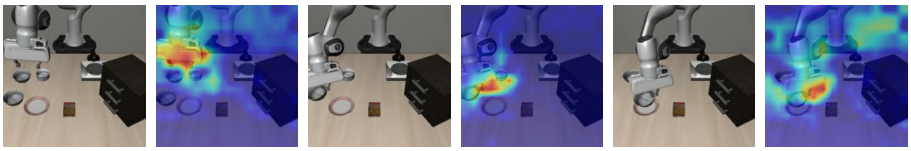

Fig. 6: Context compressor selects task-relevant, compact information. Visualization of image-compressor attention on selected frames (3/6/8), showing that the attended image regions vary across time. In-context demonstration frames and their corresponding attention maps are shown interleaved.

context demonstration. For visualization, we average the attention weights over all 32 queries. The subsampled in-context demonstration has length 8, and we show selected frames (3/6/8) in Fig. 6. We observe that the learnable queries attend to task-relevant regions according to task progress. Across time, the attention pattern changes with the task phase: before grasping, it concentrates on the gripper; after the object is grasped, it shifts to the object; and during placing, it focuses on the plate at the target location.

7 Conclusion

In this paper, we propose ContextFlow, an in-context imitation learning framework for robotics based on conditional flow matching. Unlike autoregressive models that predict discrete action tokens sequentially, ContextFlow predicts continuous actions in parallel conditioned on both the current observation and in-context demonstrations. Our empirical results suggest that flow matching appears beneficial for generalizing to unseen task configurations, where continuous parallel action prediction helps alleviate compounding error. We further find that multimodal context compressors reduce spatial and temporal redundancy in demonstrations and further improve performance. However, the generalization demonstrated in this work is mainly limited to unseen objects and spatial arrangements within tasks that share similar underlying primitives, rather than to fundamentally different task types. Scaling up the diversity and quantity of training data may further improve generalization to broader task distributions. Overall, these empirical findings highlight flow-based continuous prediction and compact context modeling as effective design choices for in-context imitation learning in robotics.

Acknowledgements

We would like to thank the anonymous reviewers for their constructive and thoughtful comments. We would like to thank Li Mi for insightful discussions. The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST) - Center of Excellence for Generative AI, under award number 5940.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al.: Qwen technical report. arXiv preprint arXiv:2309.16609 (2023)
3. Beyer, L., Steiner, A., Pinto, A.S., Kolesnikov, A., Wang, X., Salz, D., Neumann, M., Alabdulmohsin, I., Tschannen, M., Bugliarello, E., et al.: Paligemma: A versatile 3b vlm for transfer. arXiv preprint arXiv:2407.07726 (2024)
4. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al.: π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164 (2024)
5. Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al.: RT-2: Vision-language-action models transfer web knowledge to robotic control. arXiv preprint arXiv:2307.15818 (2023)
6. Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., et al.: RT-1: Robotics transformer for real-world control at scale. arXiv preprint arXiv:2212.06817 (2022)
7. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
8. Di Palo, N., Johns, E.: Keypoint action tokens enable in-context imitation learning in robotics. In: RSS (2024)
9. Du, N., Huang, Y., Dai, A.M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A.W., Firat, O., et al.: Glam: Efficient scaling of language models with mixture-of-experts. In: International conference on machine learning. pp. 5547–5569. PMLR (2022)
10. Duan, Y., Andrychowicz, M., Stadie, B.C., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., Zaremba, W.: One-shot imitation learning. In: NeurIPS. vol. 30, pp. 1088–1099 (2017)
11. Fu, L., Huang, H., Datta, G., Chen, L.Y., Panitch, W.C.H., Liu, F., Li, H., Goldberg, K.: In-context imitation learning via next-token prediction. arXiv:2408.15980 (2024)
12. Gemma Team, Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M.S., Love, J., et al.: Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295 (2024)
13. Hu, E.J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (2022), `https://openreview.net/forum?id=nZeVKeeFYf9`
14. Jaegle, A., Borgeaud, S., Alayrac, J.B., Doersch, C., Ionescu, C., Ding, D., Koppula, S., Zoran, D., Brock, A., Shelhamer, E., et al.: Perceiver io: A general architecture for structured inputs & outputs. arXiv preprint arXiv:2107.14795 (2021)
15. Jaegle, A., Gimeno, F., Brock, A., Vinyals, O., Zisserman, A., Carreira, J.: Perceiver: General perception with iterative attention. In: International conference on machine learning. pp. 4651–4664. PMLR (2021)
16. Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., Fan, L.: Vima: General robot manipulation with multimodal prompts. In: ICML (2023)

17. Kim, M.J., Finn, C., Liang, P.: Fine-tuning vision-language-action models: Optimizing speed and success. arXiv preprint arXiv:2502.19645 (2025)
18. Kim, M.J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al.: Openvla: An open-source vision-language-action model. arXiv preprint arXiv:2406.09246 (2024)
19. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In: International conference on machine learning. pp. 19730–19742. PMLR (2023)
20. Li, X., Thickstun, J., Gulrajani, I., Liang, P.S., Hashimoto, T.B.: Diffusion-lm improves controllable text generation. Advances in neural information processing systems **35**, 4328–4343 (2022)
21. Lipman, Y., Chen, R.T., Ben-Hamu, H., Nickel, M., Le, M.: Flow matching for generative modeling. arXiv preprint arXiv:2210.02747 (2022)
22. Lipman, Y., Havasi, M., Holderrieth, P., Shaul, N., Le, M., Karrer, B., Chen, R.T., Lopez-Paz, D., Ben-Hamu, H., Gat, I.: Flow matching guide and code. arXiv preprint arXiv:2412.06264 (2024)
23. Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., Stone, P.: Libero: Benchmarking knowledge transfer for lifelong robot learning. Advances in Neural Information Processing Systems **36**, 44776–44791 (2023)
24. Liu, Q.: Rectified flow: A marginal preserving approach to optimal transport. arXiv preprint arXiv:2209.14577 (2022)
25. Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., Zhu, J.: Rdt-1b: a diffusion foundation model for bimanual manipulation. arXiv preprint arXiv:2410.07864 (2024)
26. Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., Zettlemoyer, L.: Rethinking the role of demonstrations: What makes in-context learning work? arXiv preprint arXiv:2202.12837 (2022)
27. Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P., Peters, J., et al.: An algorithmic perspective on imitation learning. Foundations and Trends® in Robotics **7**(1-2), 1–179 (2018)
28. Pertsch, K., Stachowicz, K., Ichter, B., Driess, D., Nair, S., Vuong, Q., Mees, O., Finn, C., Levine, S.: Fast: Efficient action tokenization for vision-language-action models. arXiv preprint arXiv:2501.09747 (2025)
29. Physical Intelligence, Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., et al.: $\pi_{0.5}$: A vision-language-action model with open-world generalization. arXiv preprint arXiv:2504.16054 (2025)
30. Pomerleau, D.A.: Alvinn: An autonomous land vehicle in a neural network. Advances in neural information processing systems **1** (1988)
31. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)
32. Shah, R., Liu, S., Wang, Q., Jiang, Z., Kumar, S., Seo, M., Martín-Martín, R., Zhu, Y.: Mimicdroid: In-context learning for humanoid robot manipulation from human play videos. arXiv preprint arXiv:2509.09769 (2025)
33. Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., Dean, J.: Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. arXiv preprint arXiv:1701.06538 (2017)
34. Sridhar, K., Dutta, S., Jayaraman, D., Lee, I.: Ricl: Adding in-context adaptability to pre-trained vision-language-action models. arXiv preprint arXiv:2508.02062 (2025)

35. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
36. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al.: Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288 (2023)
37. Vosylius, V., Johns, E.: Instant policy: In-context imitation learning via graph diffusion. arXiv:2411.12633 (2024)
38. Vuong, A.D., Vu, M.N., An, D., Reid, I.: Action tokenizer matters in in-context imitation learning. arXiv:2503.01206 (2025)
39. Wang, L., Yang, N., Wei, F.: Learning to retrieve in-context examples for large language models. arXiv preprint arXiv:2307.07164 (2023)
40. Wen, J., Zhu, Y., Li, J., Zhu, M., Tang, Z., Wu, K., Xu, Z., Liu, N., Cheng, R., Shen, C., et al.: Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. IEEE Robotics and Automation Letters (2025)
41. Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., Gan, C.: Prompting decision transformer for few-shot policy generalization. In: international conference on machine learning. pp. 24631–24645. PMLR (2022)
42. Yin, Y., Wang, Z., Sharma, Y., Niu, D., Darrell, T., Herzig, R.: In-context learning enables robot action prediction in llms. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2025), arXiv:2410.12782
43. Zhai, X., Mustafa, B., Kolesnikov, A., Beyer, L.: Sigmoid loss for language image pre-training. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 11975–11986 (2023)
44. Zhang, X., Boularias, A.: One-shot imitation learning with invariance matching for robotic manipulation. arXiv preprint arXiv:2405.13178 (2024)
45. Zhao, T.Z., Kumar, V., Levine, S., Finn, C.: Learning fine-grained bimanual manipulation with low-cost hardware. RSS (2023)