RICL: Adding In-Context Adaptability to Pre-Trained Vision-Language-Action Models

Kaustubh Sridhar[1], **Souradeep Dutta**[2], **Dinesh Jayaraman**[1], **Insup Lee**[1]
[1]University of Pennsylvania, [2]University of British Columbia
ksridhar@alumni.upenn.edu

Abstract: Multi-task "vision-language-action" (VLA) models have recently demonstrated increasing promise as generalist foundation models for robotics, achieving non-trivial performance out of the box on new tasks in new environments. However, for such models to be truly useful, an end user must have easy means to teach them to improve. For language and vision models, the emergent ability to perform in-context learning (ICL) has proven to be a versatile and highly useful interface to easily teach new tasks with no parameter finetuning. Unfortunately, VLAs pre-trained with imitation learning objectives do not naturally acquire ICL abilities. In this paper, we demonstrate that, with the right finetuning recipe and a small robot demonstration dataset, it is possible to inject in-context adaptability *post hoc* into such a VLA. After retraining for in-context learning (RICL), our system permits an end user to provide a small number (10-20) of demonstrations for a new task. RICL then fetches the most relevant portions of those demonstrations into the VLA context to exploit ICL, performing the new task and boosting task performance. We apply RICL to inject ICL into the π_0-FAST VLA, and show that it permits large in-context improvements for a variety of new manipulation tasks with only 20 demonstrations per task, without any parameter updates. When parameter updates on the target task demonstrations is possible, RICL finetuning further boosts performance. We release code and model weights for RICL-π_0-FAST alongside the paper to enable, for the first time, a simple in-context learning interface for new manipulation tasks[1].

Keywords: Vision-Language-Action (VLA) models, In-Context Learning (ICL), Retrieval-Augmenetd Generation (RAG)

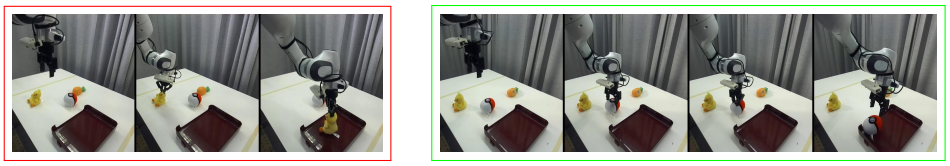
1 Introduction

Robot learning is undergoing a transformative moment with the emergence of the first generation of general-purpose Vision-Language-Action (VLA) models, capable of performing a wide spectrum of robotic tasks — a development with profound practical implications. Such models [1, 2, 3, 4, 5, 6, 7, 8] could address persistent challenges in robotics, including data scarcity, robustness, and generalization.

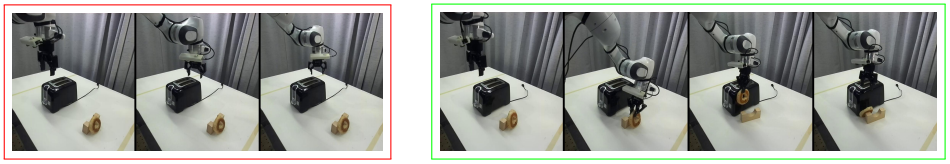
A natural point of comparison for these VLAs is large language models (LLMs). One important factor in the widespread adoption of LLMs today is that they appear to be able to quickly learn new tasks, simply through providing a few examples as "context" alongside the query, with no parameter tuning. This capability, called in-context learning (ICL) [9], emerges naturally in LLMs pre-trained for next-token prediction, due to the nature of web text data. Even better, one need not even manually provide these few examples. Instead, a retrieval mechanism could automatically fetch the most relevant data from a large corpus and place them into the LLM context. This retrieval-augmented generation (RAG) mechanism is widely adopted as a versatile interface to improve a base LLM [10].

[1] Website: https://ricl-vla.github.io

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



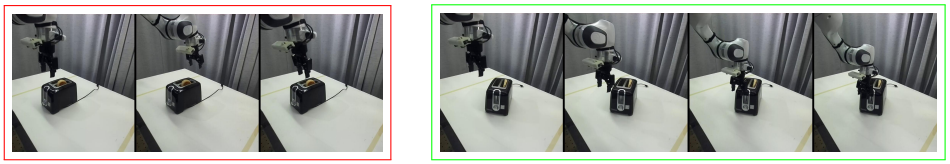
(a) Task: "pick up the poke ball and put it in the tray". π_0-FAST-DROID **[L]** picks up the distractor (duck) instead (language grounding issue). RICL-π_0-FAST-DROID **[R]** actually moves the unseen object (pokeball) with only RAG and ICL.



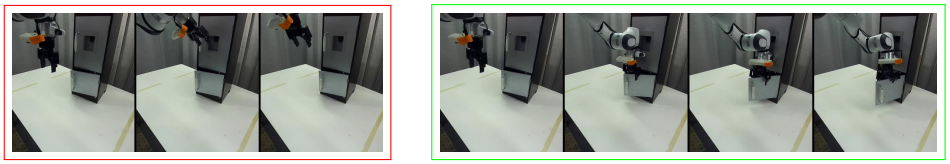
(b) Task: "pick up the bagel and put it in the toaster". π_0-FAST-DROID **[L]** aimlessly wanders and cannot figure out the grasp or motion (adaptation issue). RICL-π_0-FAST-DROID **[R]** almost completes the task (only with RAG and ICL) but drops the unseen object (bagel) at the end of the novel motion–a combination of an unfamiliar grasp at its rim, its unique initial vertical position, and the twist-and-lift motion.

(c) Task: "move the idli plate to the sink". In π_0-FAST-DROID's best test rollout shown here, it still struggles with the grasp and motion for this novel object (adaptation issue) or moves the apple (distractor) instead (language grounding issue). RICL-π_0-FAST-DROID can perform the novel motion (gripper in depressions) on the unseen object (idli plate) in this new scene with new camera positions/orientations and with lighting changes (which is different from the table where the priming demonstrations were collected).

(d) Task: "use the squeegee to clean the counter". π_0-FAST-DROID oscillates without success. It gets close, but it cannot figure out the grasp or the motion (adaptation issue). RICL-π_0-FAST-DROID adapts to novel object (squeegee) and motion (part lifting, part dragging) in the new scene. Notice the pellets dropping into sink showing contact with the surface.



(e) Task: "push the lever on the toaster". π_0-FAST-DROID **[L]** aimlessly wanders. It cannot figure out the precise location or the grasp (adaptation to a variant of training object issue). RICL-π_0-FAST-DROID **[R]** completes the precise task, only with RAG and ICL, and with elicited latent actions not in the retrieval data (more information in Section 6). This long-tail task appears infrequently in the DROID dataset.



(f) Task: "open the door of the bottom shelf". π_0-FAST-DROID **[L]** aimlessly wanders. It cannot figure out the motion to adjust or avoid the top door acting as obstacle (adaptation issue). RICL-π_0-FAST-DROID **[R]** completes the task, with this variant of a seen object (this particular shelf) and novel motion (precise door opening adjusting for the obtructing top shelf), only with RAG+ICL. This is also a long-tail task.

Figure 1: Qualitative comparison between π_0-FAST-DROID **[L]** and RICL-π_0-FAST-DROID **[R]**, with 20 task specific demonstrations for RAG and ICL, on new tasks, including novel objects, motions, and scenes. Additional comparisons can be found in Figure 8 in Appendix D.

Unfortunately, VLAs are trained with imitation learning objectives on relatively narrow demonstration datasets. As one would expect, this does not naturally produce any in-context learning abilities. This means that "improving" a pre-trained VLA today means tuning its parameters on a new demonstration dataset [2]. To make it possible for an end user to easily improve a VLA with no parameter tuning, we ask the following question:

How can we inject in-context learning abilities into a pre-trained VLA?

Once this is done, we should be able to painlessly boost the VLA's performance on any task, including handling unseen objects, novel motions, and new scenes that don't exist in the VLA training data.

Our solution is to ***r**etrain* for **i**n-**c**ontext **l**earning (RICL, pronounced "rickle"). RICL borrows from prior recipes [11, 12] to train generalist models for in-context learning and RAG. In particular, while REGENT [12] trained generalist game-playing agents from scratch, RICL uses this approach to instead post-train an off-the-shelf VLA priming it to use its context effectively. The resulting RICL-VLA can improve the base VLA's performance for any target tasks without a single gradient update, instead adapting purely through retrieval-augmentation and in-context learning, with only 10-20 demonstrations in its retrieval buffer. We demonstrate this on various manipulation tasks depicted in Figure 1 where a state-of-the-art (SOTA) VLA fails but our RICL-VLA adapts simply via RAG and ICL. We further find that it is possible to get even better task performance by "finetuning like you pretrain" [13]: we optimize the RICL objective on the same demonstrations as used above for ICL, and get large performance boosts.

2 Related Work

Training VLAs and multi-task generalist agents: There has been a spate of work in recent years on training multi-task agents in simulated settings like games [14, 12, 15, 16, 17, 18] and in recent months on training VLAs for robotics [1, 2, 3, 4, 6, 7, 5, 8]. To our knowledge, there are only three prior attempts to train general agents with in-context learning (ICL) capabilities [16, 12, 18], none for general-purpose robotics. This is the focus of RICL: we show how to post-train a pre-trained VLA to effectively learn in-context.

In-context learning for robotics: The in-context learning abilities of large language models (LLMs) and vision-language models (VLMs) have already been found to be very useful in robotics: with suitable representations (such as keypoints or code), these models can in-context learn imitation policies [19, 20, 21, 22] or value functions [23, 24]. But, using LLMs and VLMs requires these methods to run completely (or mostly) open-loop [19, 20, 21] and using only LLMs makes them lose significant visual information [19, 21]. Both these drawbacks affect their ability to adapt.

3 Background on ICL, RAG, and π_0 VLAs

In-Context Learning (ICL) is the property of sequence to sequence models that allows them to predict an output (such as an action $\hat{a}_t$) for a new input (such as state s_t) given a few examples of input-output pairs in the context (such as state-action pairs $\{(s', a'), (s'', a''), ...\}$). The input to the model is the concatenation of the context and the new input.

Retrieval-Augmented Generation (RAG) refers to the strategy commonly used to help LLMs predict an answer for a query. This is achieved by obtaining the information necessary for the answer by searching through a dataset and then placing said information in the context of an LLM.

Of the three methods for embodied agents mentioned in Section 2 that learn to in-context learn, MTT [16] and ICRT [18] place a few complete demonstrations in their context while REGENT [12] retrieves select states and actions (from the same few complete demonstrations) to place in its context. The RAG+ICL method employed by REGENT outperforms the former ICL method across held-out games and held-out simulated robotics tasks. REGENT demonstrates that combining the two i.e. retrieving specific examples to place into context from a demonstration dataset offers a computationally less intensive, higher performing alternative to directly placing demonstrations in context. We adopt this idea in RICL.

π_0-FAST [1] is a state-of-the-art auto regressive VLA model that takes images, language instruction, and proprioceptive state as input and predicts actions. It can be deployed (in a variety of scenes) zero-shot on robot embodiments that are a part of its training data and few-shot after finetuning on new robot embodiments. **π_0-FAST-DROID** [1] is a VLA that was created by further finetuning π_0-FAST on the large DROID dataset [25]. The DROID dataset was collected with the Franka DROID platform, shown in Figure 3, across many research labs (additional details in Appendix B).

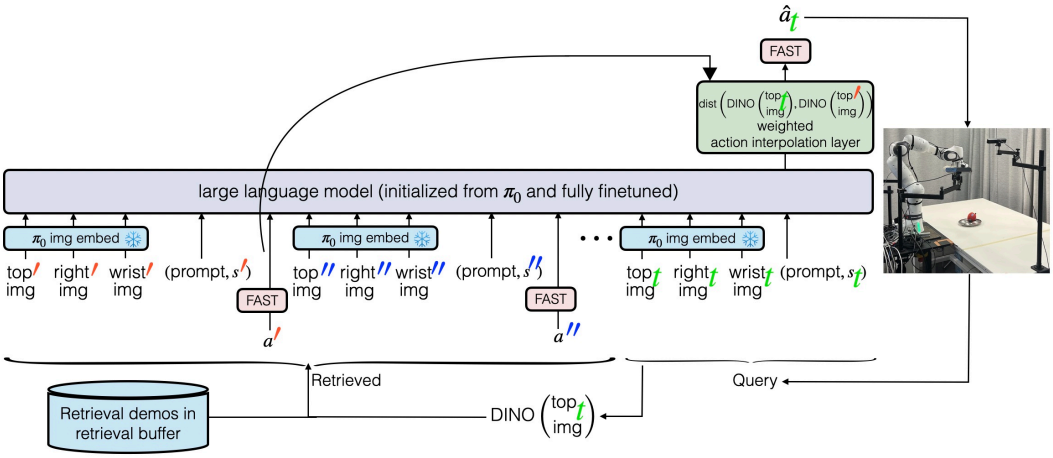


Figure 2: Architecture of RICL-VLAs, specifically that of RICL-π_0-FAST.

4 RICL and creating in-context learning capable RICL-VLAs

This work aims to combine the best of both worlds from Section 3–*i.e.*, it aims to quickly convert a VLA that can be generally deployed like π_0-FAST-DROID into one that also has in-context adaptation capabilities like REGENT. Once an in-context learning capable VLA has been created, "teaching" it to improve its performance on a new task is as simple as downloading the model, collecting a few demonstrations, and providing them as a retrieval dataset. Then, the in-context learning capable VLA should instantly have much better success rates on this new task than the baseline VLA.

Re-training for In-Context Learning (RICL): RICL enables the aforementioned conversion of a pre-trained VLA to a in-context learning capable-VLA (that we call a RICL-VLA). In RICL, a VLA is post-trained on sequences of query images/states and many images, states, actions, and action obtained from the retrieval demonstrations as depicted in Figure 2. The query information at time t consists of three images (top image$_t$, side image$_t$, wrist image$_t$), a language prompt describing the task, and proprioceptive state s_t. We use the term "query" following terminology from RAG for LLMs. The retrieved neighbors also consist of three images, the same text prompt, proprioceptive state, and action chunk (*i.e.* an array of actions over many time steps). The retrieved information is placed in the context with the closest neighbor (to the query) on the left and farther away neighbors towards the right. The closest neighbor's images, states, and actions is represented with a single $\prime$, the second closest with a double $\prime\prime$ and so on (see Figure 2). This finetuning utilizes a few "priming" demonstrations. These demonstrations are called "priming" demonstrations since their role is to prime the VLA to use its context effectively. Further, as depicted in Figure 2, only the LLM is finetuned during RICL while the image encoder is kept frozen. RICL-VLAs perform retrieval by embedding only the top query image with an off-the-shelf DINO-v2 [26] image encoder and comparing it with the embeddings of top images of the demos in the retrieval buffer with an ℓ_2 distance metric.

Like REGENT [12], the predicted action $\hat{a}_t$ involves a distance-weighted interpolation between the action tokens of the closest retrieved action $a\prime$ and the final output of the large language models. We refer to this as the action interpolation layer and depict it within the green box above the LLM in Figure 2. This distance corresponds to the distance between the DINO embeddings of the query top image and the closest retrieved top image. The action interpolation layer assumes a maximum number of action tokens numbering N_{act} and combines the one-hot encoding of each token of a' with the corresponding token output by the LLM $\pi_\theta(\text{retrieved, query})$ as follows:

$$\pi^{\theta}_{\texttt{RICL-VLA}}(\text{retrieved, query}) = e^{-\lambda d}\ \text{one-hot}(a') + (1 - e^{-\lambda d})\sigma\left(\pi_\theta(\text{retrieved, query})\right) \quad (1)$$

where σ represents the Softmax function and d denotes the ℓ_2 distance between the DINO embeddings of top image$_t$ and its nearest neighbor top image$'$. The RICL-VLA performs the above interpolation for each of the N_{act} tokens. These tokens are then detokenized by the FAST tokenizer to obtain an action chunk that can be executed on the robot.

Unlike the process to train REGENT [12], RICL on the other hand, only predicts and minimizes the cross-entropy loss over the query (prompt, s_t) tuple and predicted action chunk during training whereas REGENT [12] predicted and minimized the loss over all retrieved and query actions.

The RICL-VLA, after having been primed to use its context in RICL, can now be deployed on a target task, which can include unseen objects and novel motions, with just a few task-specific demonstrations for RAG and ICL, and without any further training on those demonstrations.

Further finetuning of a RICL-VLA: If a RICL-VLA is further finetuned on those target task demonstration–that it was only retrieving from and throwing into its context previously–it can significantly improve its performance, outperforming a VLA directly vanilla finetuned on those unseen task demonstrations. This finetuning process on the few task-specific demonstrations is done exactly like RICL– *i.e.*, a retrieval-augmented finetune of the RICL-VLA (which has the action interpolation layer) with the same objective of minimizing the cross-entropy loss over the query (prompt, s_t) tuple and predicted action chunk. At deployment, the finetuned RICL-VLA still retrieves from the same data that it is finetuned with, *i.e.* no extra data (hyperparameters in Appendix C).

5 Experimental Setup

Training (*a.k.a.*, Priming) data for RICL: We collect 20 demonstrations with the Franka DROID platform (see Figure 3), randomizing the initial position of the primary object, in 20 pick and place tasks (total 400 demonstrations). The exact list of tasks, an image of all the objects used, and more details about the platform are in Appendix C. We perform RICL– starting from the weights of π_0-FAST-DROID, fully finetuning its LLM, and keeping its image encoder frozen– with the hyperparameters detailed in Appendix C. We call the model obtained after three epochs of training as RICL-π_0-FAST-DROID.

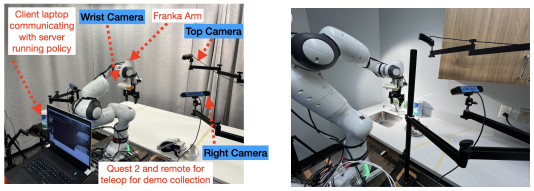


Figure 3: [LEFT] Our Franka DROID setup, annotated. [RIGHT] Franka DROID, including the top camera and right camera, moved to a new scene (kitchen sink).

In each task, for each demonstration, and for each state in that demonstration, we use that state as the query and retrieve four neighbors from the other 19 demonstrations to create training input sequences. In this way, we end up training the model in the same way we would deploy the model. We collect data as detailed above since such a dataset is not available.

Evaluation tasks with unseen objects, novel motions, different scenes: We evaluate all methods on the following tasks, which involve a task-relevant unseen object and/or a completely novel motion in two different scenes (tabletop and kitchen sink). "Unseen" here means that these objects and motions are not in either the RICL priming data or the DROID data [25]. We checked the latter by searching over the DROID dataset's language annotations (and some recordings when language was not adequate). First, we start with a simple task that primarily tests language grounding.

- (pokeball) "pick up the pokeball and put it in the tray": has an unseen object (pokeball) with a couple of distractors on the table.

Next, we test on simple tasks that test both language grounding and adaptation to novel motions.

- (idliplate) "move the idli plate to the left": has an unseen object (idli plate) with a apple (distractor) sitting on the plate. It also requires the robot to do an unfamiliar grasp to move the uniquely shaped plate with depressions to the right.

- (squeegee) "move the squeegee to the right and try to drag it": has an unseen object (squeegee). It also requires the robot to do a novel motion of slightly lifting the handle of the squeegee while keeping its rubber on the table to drag it across the table.

We then test on versions of the above two simple tasks in a new scene (kitchen sink area) with new camera positions/orientations (no calibration necessary), new lighting, and new distractors.

- (`sink-idliplate`) "move the idli plate to the sink": all of the previous challenges & a new scene.
- (`sink-squeegee`) "use the squeegee to clean the counter": all of the challenges of the previous task with pellets to be cleaned up on the counter in the new scene.

We also test on the long-tail of the training task distribution. These tasks consist of variations of objects in the DROID dataset. These object classes appear infrequently in the dataset.

- (`toaster`) "push the lever on the toaster": has a different version of an object (a particular toaster brand) in the DROID dataset. This task tests the long tail of the training task distribution. It also requires a precise placement and movement to push the lever down.
- (`door`) "open the door of the bottom shelf": has a different version of an object (a particular shelf) in the DROID dataset. This task also is in the long tail. It requires a novel and precise motion that can handle the large top shelf acting as an obstacle when reaching the bottom door's handle.

Finally, we test on a longer horizon task that is a composition of many simple tasks.

- (`bagel`) "pick up the bagel and put it in the toaster": has an unseen object (bagel). It also requires the robot to do a composite novel motion–an unfamiliar grasp on the edge of the bagel, the twist-and-lift motion, and placing in the slot.

Retrieval data in evaluation tasks: In all the evaluation tasks, we collect 20 demonstrations, randomizing the initial position of the primary object, for `RICL`-π_0-FAST-DROID to retrieve from and throw in its context to adapt to the task.

Evaluation metrics and comprehensive randomization: We collect 10 test rollouts for all methods on all evaluation tasks with randomly chosen initial positions and orientations in each rollout. We set these intial positions and orientations all across the table. The distractors, if any, are kept approximately in the same region of the table but they are also not fixed in place. We calculate the success of the full tasks, in addition to tracking intermediate checkpoints for a better understanding of the progress of each method on each task.

Baselines and ablations: We compare `RICL`-π_0-FAST-DROID with vanilla π_0-FAST-DROID on all tasks. We also compare with 'Retrieve and play', a 1 nearest neighbor baseline from [12], which simply outputs the first retrieved action a'. We also compare with a trained-from-scratch Diffusion Policy baseline. We perform these two comparisons on the simpler evaluation tasks (`pokeball`, `idliplate`, `squeegee`). Upon observing their low success rates, we leave them out for the more complex tasks. In the tasks performed in a new scene (`sink-idliplate`, `sink-squeegee`), we aim to test `RICL`'s ability to retain π_0's helpful scene-generalization capabilities while adapting in-context to a new task and hence only test these two methods. We also ablate the number of demonstrations used by each method in the `idliplate` task.

Further finetuning: We further finetune `RICL`-π_0-FAST-DROID on each evaluation task on the 20 demonstrations collected for retrieval. For comparison, we also further (vanilla) finetune π_0-FAST-DROID on these same 20 demonstrations in each task.

6 Experimental Evaluation

Generalization to unseen objects and novel motions and new scenes: We plot the quantitative results across tasks and methods in Figure 4. We observe that `RICL`-π_0-FAST-DROID outperforms π_0-FAST-DROID, especially in earlier checkpoints of the task, but also in overall task success. In aggregate, across all evaluated tasks, π_0-FAST-DROID obtains a complete task success rate of 2.5% and a checkpoint completion rate of up to 21.25%. On the other hand, `RICL`-π_0-FAST-DROID obtains a significantly improved complete task success rate of 31.25% and a checkpoint completion rate of up to 83.75%.

We particularly note that `RICL`-π_0-FAST-DROID has significantly improved language grounding to move towards the unseen objects just based on contextual information. More importantly, `RICL`-

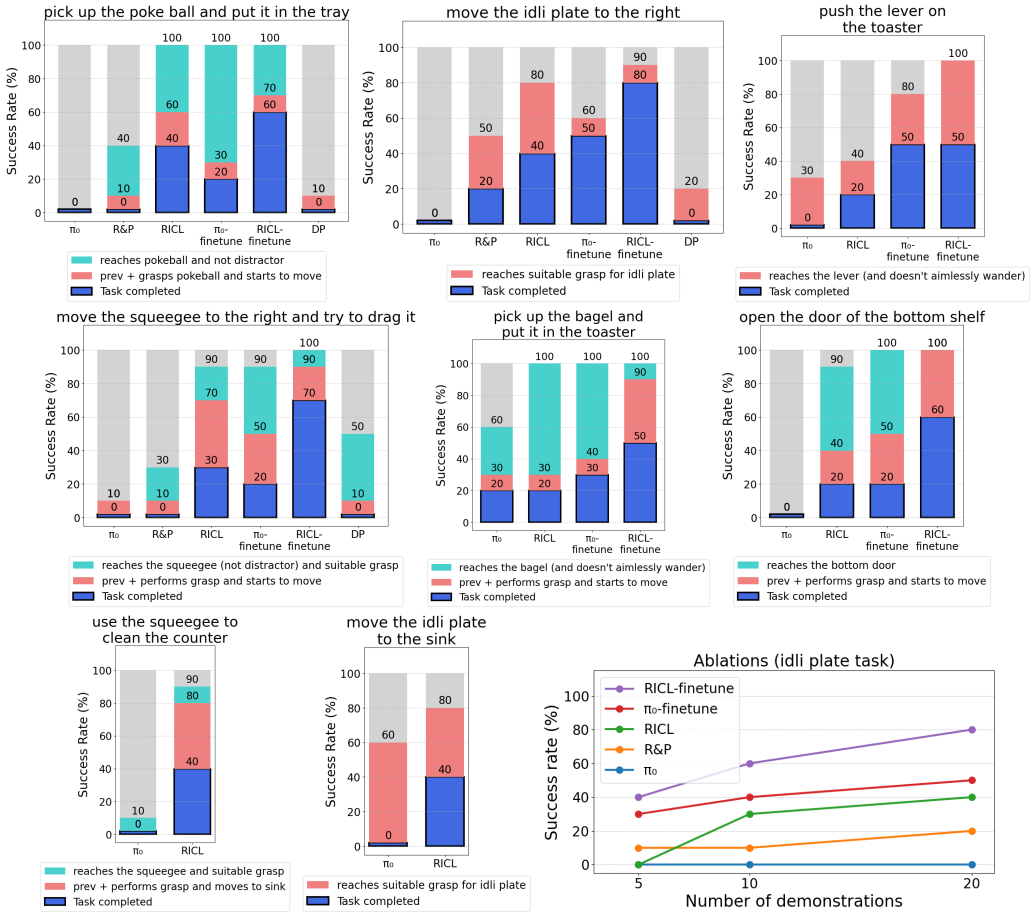


Figure 4: Success rates of 10 test rollouts from various methods across various tasks represented by stacked bar plots. The lowest bar (dark blue) in each stacked column represents full task success rate, and other bars are the success rates for reaching earlier waypoints. Gray regions represent the fraction of runs that did not even reach the first waypoint for the task. We note that π_0 refers to π_0-FAST-DROID and RICL to `RICL`-π_0-FAST-DROID in the plots. We also plot the performance of various methods vs the number of demonstration in the `idliplate` task on the bottom right.

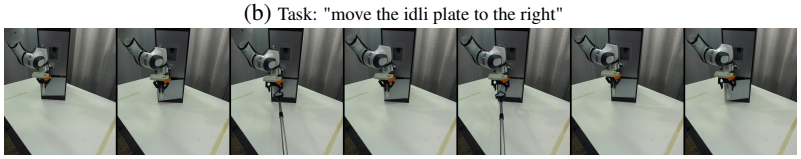
π_0-FAST-DROID also overcomes the adaptation issue faced by π_0-FAST-DROID. Where π_0-FAST-DROID struggles with grasps and motions, `RICL`-π_0-FAST-DROID demonstrates the ability to infer novel grasps and motions from its context as evidenced in six tasks (both `squeegee` tasks, `sink-idliplate`, `bagel`, `toaster`, and `door`). We plot the qualitative results depicting key test rollouts and behaviors of π_0-FAST-DROID and `RICL`-π_0-FAST-DROID in Figure 1 and Figure 8. We also provide side-by-side comparisons and detailed explanations of rollouts in the same Figures.

Unexpectedly, we observe in some tasks that `RICL`-π_0-FAST-DROID seems to predict and execute action sequences that are not like the motions in the retrieval dataset. For example, in `idliplate`, `RICL`-π_0-FAST-DROID moves to the left of the idli plate, closes its gripper and pushes the plate to the right. But, all 20 demonstrations in the retrieval buffer were collected with the motion of dipping the gripper into the depressions and moving the plate to the right. Hence, `RICL`-π_0-FAST-DROID has seemingly elicited latent actions or knowledge to accomplish this task (also seen in `toaster`).

Significantly improved performance after further finetuning the `RICL`-VLA: We observe a significant improvement in performance after further finetuning `RICL`-π_0-FAST-DROID on each task's 20 demonstrations. In aggregate, across all evaluated tasks, π_0-FAST-DROID-finetuned obtains a complete task success rate of 31.67%, while `RICL`-π_0-FAST-DROID-finetuned obtains a complete task success rate of 61.67%. In fact, we not only see that the `RICL`-VLA with finetuning is significantly better than the base VLA with finetuning (at almost double the aggregate performance).

(a) Task: "pick up the poke ball and put it in the tray"

(b) Task: "move the idli plate to the right"



(c) Task: "open the door of the bottom shelf"

Figure 5: Qualitative visualization of the reactivity and robustness of RICL-π_0-FAST-DROID-finetuned on 20 task-specific demonstrations in a dynamic test rollout. In the above, a human randomly perturbs and displaces the primary object during the test rollout. Additional results can be found in Figure 9 in Appendix D.

But we also observe comparable performance in complete task success rate, in aggregate, between the RICL-VLA (at 31.25%), which only uses RAG and ICL, and the base VLA with finetuning on the target task data (at 31.67%).

We hypothesize that further finetuning our VLA is significantly better than doing so with the base VLA simply because our VLA can use all of its capacity to focus on interpolating amongst the retrieved images, states, and actions to predict a new action and does not have to memorize any data. This is in line with the observed parameter-efficiency & performance advantages of RAG LLMs [27].

We qualitatively demonstrate the reactivity and robustness of RICL-π_0-FAST-DROID by randomly perturbing and displacing objects during a test rollout as shown in Figure 5 and Figure 9.

Ablating the number of retrieval/finetuning demonstrations: We plot the ablation results, ablating the number of demonstrations used in the retrieval buffer or for finetuning, for `idliplate` in the bottom right of Figure 1. We found that too few demonstrations (such as 5) results in RICL-π_0-FAST-DROID starting to behave like π_0-FAST-DROID to the extent where in one test rollout, it too moves the apple instead of the idli plate. This demonstrates the requirement for atleast 10 demonstrations. Also, from this figure, we conclude that more retrieval demonstrations help RICL-π_0-FAST-DROID improve, towards catching up with π_0-FAST-DROID-finetuned. We also see that RICL-π_0-FAST-DROID-finetuned is significantly better than π_0-FAST-DROID-finetuned at every number of demonstrations.

No loss-of-capabilities results: One might wonder: does RICL post-training come at the cost of losing the ability in the base VLA to perform without any retrieval data? To evaluate this, we test RICL-π_0-FAST-DROID with randomly chosen priming demonstration in the retrieval buffer, rather than any meaningful task-specific demonstrations, on three tasks: "move the can to the tray", "pick up the marker and put it in the tray", and "place the apple next to the can". It obtains an 80% success rate, just like π_0-FAST-DROID, demonstrating that RICL has not led to any loss of capabilities.

7 Conclusions and Future Work

We have demonstrated how RICL can be used to convert a VLA to a RICL-VLA that can use its context to adapt to completely new tasks, including unseen objects and novel motions, with just RAG and ICL. We found the RICL-VLA to even have comparable performance, in aggregate, with the base VLA finetuned on target task data. We have also demonstrated a significant boost in performance when a RICL-VLA is further finetuned on task-specific demonstrations. In future work, we believe that scaling up RICL in both the number of priming demonstrations and parameter size will further boost the performance of the in-context learning capable RICL-VLA.

8 Limitations

We find that RICL does not typically enable learning any tasks that are too far beyond those that the base VLA can perform. This is why in this paper we primarily focus on pick and place tasks, the primary strength of the π_0 VLAs used in this work. Further, while RICL-VLAs, unlike vanilla VLAs, can adapt to new tasks in-context, they can still benefit from improved performance. We believe that scaling up RICL in both the number of priming demonstrations/tasks and number of parameters can help with this issue in future work. Another limitation of RICL-VLAs remains their need for a few teleoperated demonstrations. Collecting these demonstrations for every new task across settings is not scalable. We believe that videos of human demonstrations (such as in [20]) can help bridge this gap. In our experiments, RICL-π_0-FAST-DROID very easily removes the issue of language grounding for new tasks and motions but struggles with generalizing to significantly novel motions such as playing a forehand with a tennis racket. We believe future work including more diverse motions, rather than just pick and place, in the RICL dataset can fix this issue.

Acknowledgments

This work was supported in part by ARO MURI W911NF-20-1-0080, NSF 2143274, NSF CAREER 2239301, NSF 2331783, and DARPA TIAMAT HR00112490421. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views the Army Research Office (ARO), the Department of Defense, or the United States Government. We also thank the anonymous reviewers for their helpful comments.

References

[1] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. (Cited on 1, 3, 13)

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. (Cited on 1, 2, 3, 13)

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. (Cited on 1, 3)

[4] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024. (Cited on 1, 3)

[5] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, et al. Robocat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023. (Cited on 1, 3)

[6] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023. (Cited on 1, 3)

[7] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. (Cited on 1, 3)

[8] G. R. Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025. (Cited on 1, 3)

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165. (Cited on 1)

[10] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023. (Cited on 1)

[11] Y. Gu, L. Dong, F. Wei, and M. Huang. Pre-training to learn in context. *arXiv preprint arXiv:2305.09137*, 2023. (Cited on 3)

[12] K. Sridhar, S. Dutta, D. Jayaraman, and I. Lee. REGENT: A retrieval-augmented generalist agent that can act in-context in new environments. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=NxyfSW6mLK. (Cited on 3, 4, 5, 6, 13)

[13] S. Goyal, A. Kumar, S. Garg, Z. Kolter, and A. Raghunathan. Finetune like you pretrain: Improved finetuning of zero-shot vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19338–19347, 2023. (Cited on 3)

[14] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022. (Cited on 3)

[15] Q. Gallouédec, E. Beeching, C. Romac, and E. Dellandréa. Jack of all trades, master of some, a multi-purpose transformer agent. *arXiv preprint arXiv:2402.09844*, 2024. (Cited on 3)

[16] S. C. Raparthy, E. Hambro, R. Kirk, M. Henaff, and R. Raileanu. Generalization to new sequential decision making tasks with in-context learning. *arXiv preprint arXiv:2312.03801*, 2023. (Cited on 3)

[17] J. Bruce, M. D. Dennis, A. Edwards, J. Parker-Holder, Y. Shi, E. Hughes, M. Lai, A. Mavalankar, R. Steigerwald, C. Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024. (Cited on 3)

[18] L. Fu, H. Huang, G. Datta, L. Y. Chen, W. C.-H. Panitch, F. Liu, H. Li, and K. Goldberg. In-context imitation learning via next-token prediction. *arXiv preprint arXiv:2408.15980*, 2024. (Cited on 3)

[19] N. Di Palo and E. Johns. Keypoint action tokens enable in-context imitation learning in robotics. *arXiv preprint arXiv:2403.19578*, 2024. (Cited on 3)

[20] G. Papagiannis, N. Di Palo, P. Vitiello, and E. Johns. R+ x: Retrieval and execution from everyday human videos. *arXiv preprint arXiv:2407.12957*, 2024. (Cited on 3, 9)

[21] Y. Yin, Z. Wang, Y. Sharma, D. Niu, T. Darrell, and R. Herzig. In-context learning enables robot action prediction in llms. *arXiv preprint arXiv:2410.12782*, 2024. (Cited on 3)

[22] S. Mirchandani, F. Xia, P. Florence, B. Ichter, D. Driess, M. G. Arenas, K. Rao, D. Sadigh, and A. Zeng. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*, 2023. (Cited on 3)

[23] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023. (Cited on 3)

[24] Y. J. Ma, J. Hejna, C. Fu, D. Shah, J. Liang, Z. Xu, S. Kirmani, P. Xu, D. Driess, T. Xiao, et al. Vision language models are in-context value learners. In *The Thirteenth International Conference on Learning Representations*, 2024. (Cited on 3)

[25] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024. (Cited on 3, 5, 13)

[26] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. (Cited on 4)

[27] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark, et al. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR, 2022. (Cited on 8)

[28] K. Sridhar, S. Dutta, D. Jayaraman, J. Weimer, and I. Lee. Memory-consistent neural networks for imitation learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=R3Tf7LDdX4. (Cited on 13)

[29] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. (Cited on 13)

[30] L.-H. Lin, Y. Cui, A. Xie, T. Hua, and D. Sadigh. Flowretrieval: Flow-guided data retrieval for few-shot imitation learning. *arXiv preprint arXiv:2408.16944*, 2024. (Cited on 13)

[31] M. Du, S. Nair, D. Sadigh, and C. Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. *arXiv preprint arXiv:2304.08742*, 2023. (Cited on 13)

[32] L. Beyer, A. Steiner, A. S. Pinto, A. Kolesnikov, X. Wang, D. Salz, M. Neumann, I. Alabdulmohsin, M. Tschannen, E. Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024. (Cited on 13)

[33] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11975–11986, 2023. (Cited on 13)

[34] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. (Cited on 13)

Appendix

A Additional Related Work

We discuss some other relevant work below.

Retrieval in robotics: Recent work on retrieval in robot learning has created surprisingly adaptive and reliable policies [28, 29, 30, 31]. Early work in VINN [29] and recent work in MCNN [28] demonstrates the capabilities of retrieval based on image embeddings to improve behavior cloning policies. Recent work in behavior [31] and flow retrieval [30] demonstrate the ability of retrieval to augment task-specific data with a subset of offline data to further improve behavior cloning policies. While RICL builds on the former use of retrieval, we believe future work that includes the latter use of retrieval could further improve RICL-VLAs that are finetuned on task-specific demonstrations.

B Additional Background Information

We provide additional background information below.

π_0-FAST: It was created by finetuning the PaliGemma Vision-Language model (VLM) [32] on a (unknown but) large number of robot trajectories from different embodiments. The PaliGemma VLM is a combination of a SigLIP image encoder [33] and the 3B parameters Gemma large language model (LLM) [34]. In particular, it differentiates itself from the similar π_0 diffusion-based VLA [2] with the use of the FAST tokenizer [1] to bring action tokens into the text token space for simple autoregressive prediction.

π_0-FAST-DROID: π_0-FAST-DROID was created by finetuning $\pi_0 - FAST$ on the DROID dataset [25]. The DROID dataset was collected with the Franka DROID platform–which consists of a Franka arm on a movable platform with cameras on the left, right, and wrist–in-the-wild, across universities and scenes and with a large variety of objects. Each episode in the DROID dataset also has proprioceptive states, actions, and language annotations. In this work, we use the DROID platform (see Figures 3 and 2) for all results and hence, we will refer to this model frequently in the rest of the paper.

REGENT [12], as discussed above, trains a transformer based multi-task policy on sequences of query and retrieved state-action pairs, *from scratch*, on 145 simulated robotics tasks and games. Given a few demonstrations in a held out simulated environment or game, REGENT retrieves state-action pairs from these demonstrations, throws it into its context, and plays the held out game.

C Additional Info on Franka DROID Setup, RICL, and RICL-π_0-FAST-DROID

Exact list of tasks for RICL: We collect 20 demonstrations in each of the following 20 tasks (for a total of about 400 demonstrations):

- 5x of "move <object> to the right" where object in [apple, orange, strawberry, the coffee pod, the cup],
- 5x of "move <object> to the left" where object in [box, cup, duck, pan, pot],
- 6x of "pick up <object> and put it in the bowl" where object in [the block, the cloth, the grapes on the waffle, the orange, the pineapple, the watermelon slice],
- 1x of "pour the contents of the mug into the bowl"
- 1x of "pick up the spoon in the bowl and put it in the other bowl"
- 1x of "put the cable in the bowl"
- 1x of "move the watermelon slice from one bowl to the other bowl"

Figure 6: **[LEFT]** Objects used during RICL, either as the primary object in the task or a distractor, to create RICL-π_0-FAST-DROID. **[RIGHT]** Unseen objects and depictions of novel motions in red arrows. Unseen tasks also include distractors from the objects used during training.

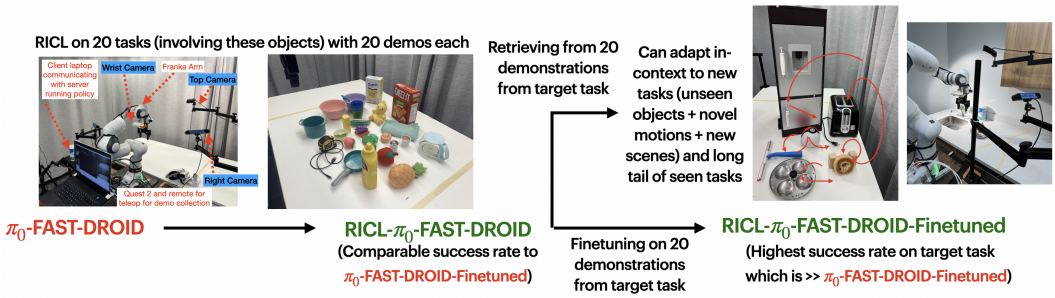


Figure 7: Overview of RICL.

the pictures of these objects and distractors used can be seen in Figure 6. We also depict the unseen objects and novel motions in the same Figure. We note that we collect data at an action frequency of 15 Hz.

Observation and action space: The proprioceptive observation consists of 7 joint angle and gripper position. The action chunks are of shape (15, 8) where 15 corresponds to the prediction horizon (which we empirically found to have better performance than 10). Each action has 7 joint velocities and 1 gripper position. All images are resized with padding to 224×224.

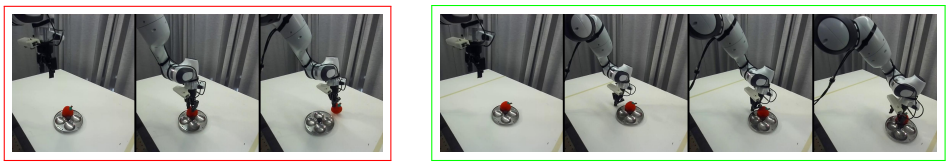
More details and key hyperparameters for RICL on priming demonstrations to create RICL-π_0-FAST-DROID: We use four retrieved groups in our context and set $\lambda = 10$. For all other hyperparameters we build on top of the openpi repository[2] and make the following key changes: using action chunks of 15 steps, training for three epochs, with a CosineDecaySchedule with 300 warmup steps, 2.5e-5 peak learning rate, 3000 decay steps, and 2.5e-6 decay learning rate. We also use a batch size of 16 and two A100 GPUs. Our detailed codebase can be found in our website.

Key hyperparameters for further finetuning RICL-π_0-FAST-DROID on each task's 20 demonstrations: We use a recipe similar to RICL but only finetune for a 1000 steps, and with the same learning rate scheduler but only for 1000 decay steps with 50 warmup steps.

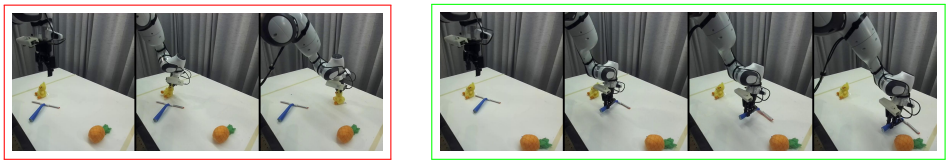
Key hyperparameters for further π_0 finetuning on each task's 20 demonstrations: We use the same number of train setps, decay steps and warmup steps in the optimizer as above but otherwise use the recommended pi0 fionetuning recipe in the openpi repository.

Key hyperparameters at deployment: For RICL-π_0-FAST-DROID, we execute 3 of the predicted 15 actions in the action chunk until a gripping action is predicted in the last action of an action

[2] https://github.com/Physical-Intelligence/openpi



(a) Task: "move the idli plate to the right". π_0-FAST-DROID **[L]** moves the apple sitting on the plate (language grounding issue). RICL-π_0-FAST-DROID **[R]** moves the unseen object (idli plate) with depressions that require an unfamiliar grasp, only with RAG and ICL. Further, it does so with elicited latent actions not in the retrieval data (see Section 6).



(b) Task: "move the squeegee to the right and try to drag it". π_0-FAST-DROID **[L]** moves the distractor (duck) instead (language grounding issue). In other rollouts, it cannot figure out the grasp or the motion (adaptation issue). RICL-π_0-FAST-DROID **[R]** moves the unseen object (squeegee) in a novel motion of partly lifting and partly dragging, only with RAG and ICL

Figure 8: Additional qualitative comparisons between π_0-FAST-DROID **[L]** and RICL-π_0-FAST-DROID **[R]**, with 20 task specific demonstrations for RAG and ICL, on new tasks, including novel objects, motions, and scenes.

chunk, at which point we switch to using 8 of the predicted 15 actions in an action chunk. Similarly, for RICL-π_0-FAST-DROID-finetuned, we execute 3 of the predicted 15 until a gripping action is predicted, at which point we switch to using 5 of the predicted 15.

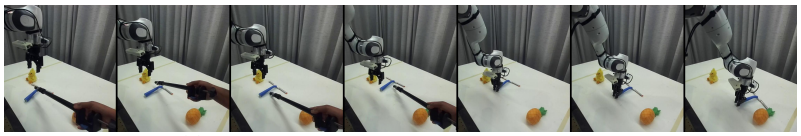
Discussion on runtime of RICL: The retrieval step occurs in less than a millisecond with the help of an existing search index library (FAISS). The 4x increase in number of tokens only results in RICL-pi0-FAST taking about 1.33x the time of pi0-FAST in a rollout. This can be observed in the videos on our website where pi0 is at 6x real time and RICL-pi0 at about 8x real time.

Discussion on RICL's performance when retrieval demonstration data is different from test conditions: During our experiments, minor changes did occur in the camera viewpoint and the condition of the scene (positions of distractors, tape on the tables, etc), and RICL did handle these minor changes. However, RICL cannot currently handle more significant changes to the camera viewpoint or scene. We believe that if RICL is trained in the priming phase with such changes then it would be more robust to such changes.

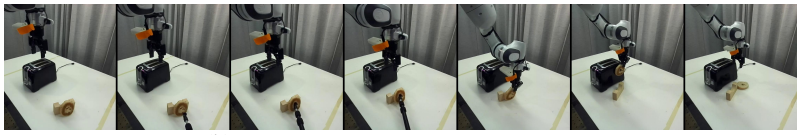
D Additional Results

We provide additional qualitative comparisons between π_0-FAST-DROID and RICL-π_0-FAST-DROID in Figure 8. We also provide additional qualitative visualizations of the reactivity and robustness of RICL-π_0-FAST-DROID-Finetuned in Figure 9.

Understanding the quality of retrieval (and hence the quality of the context): To understand the quality of retrieval, we depict retrieved top images and their corresponding side and wrist images for key states in Figure 10 for the poke ball task. We find that our simple retrieval mechanism can find similar states with close initial positions and RICL-π_0-FAST-DROID interpolates amongst these similar states and actions to predict a suitable action.



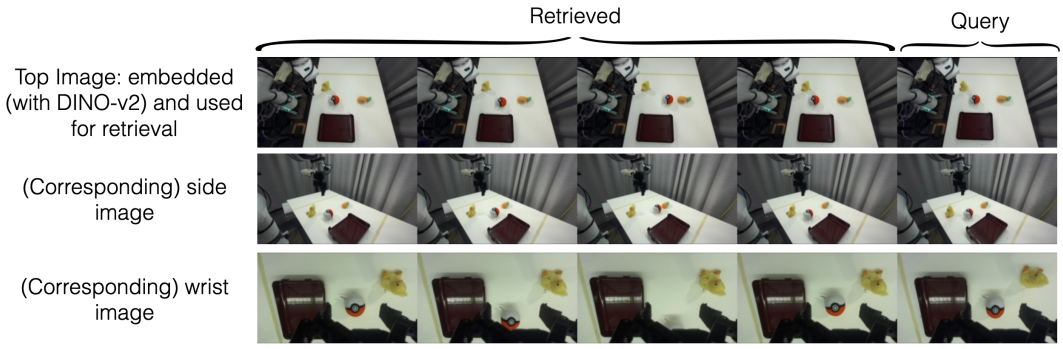
(a) Task: "move the squeegee to the right and try to drag it"



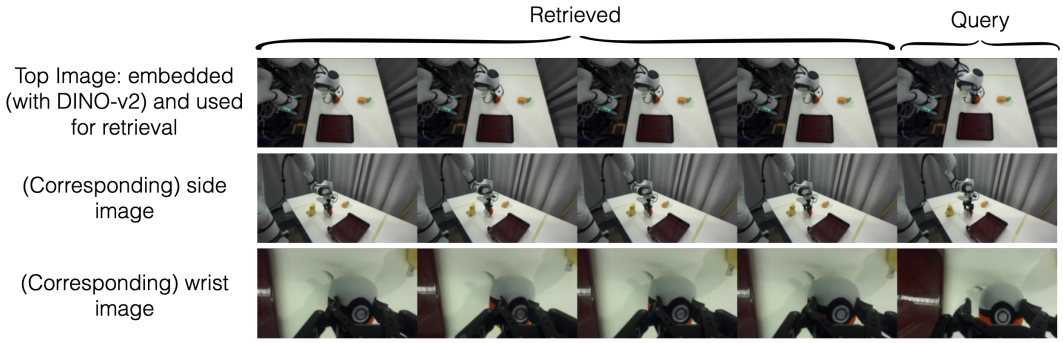
(b) Task: "pick up the bagel and put it in the toaster"

(c) Task: "push the lever on the toaster"

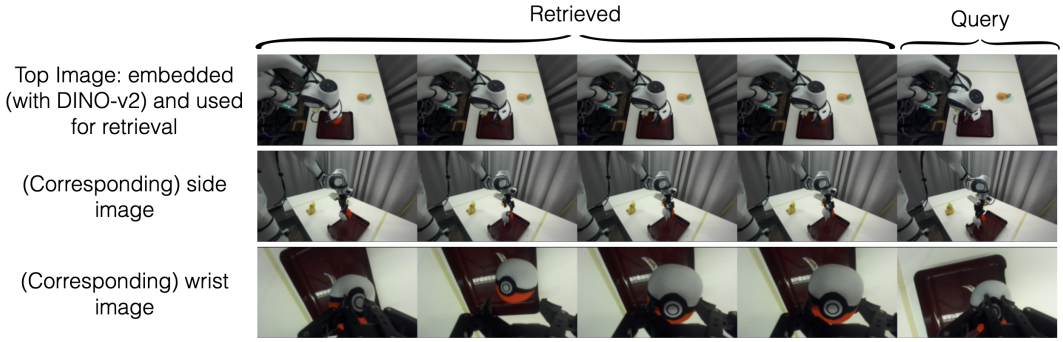
Figure 9: Additional qualitative visualizations of the reactivity and robustness of RICL-π_0-FAST-DROID-finetuned on 20 task-specific demonstrations in a dynamic test rollout. In the above, a human randomly perturbs and displaces the primary object during the test rollout.



(a) First state of the test rollout



(b) An intermediate state of the test rollout when the Franka arm starts to grasp the pokeball



(c) Last state of the test rollout at which an action chunk was predicted

Figure 10: Visualization of Query and Retrieved images at three states during a `RICL`-π_0-FAST-DROID test rollout on the `pokeball` task.