Grounding Multimodal Large Language Models in Actions

Andrew Szot[1,2] **Bogdan Mazoure**[1] **Harsh Agrawal**[1] **Devon Hjelm**[1,3]
Zsolt Kira[2] **Alexander Toshev**[1]
[1] Apple, [2] Georgia Tech, [3] Mila
a.szot@apple.com, toshev@apple.com

Abstract

Multimodal Large Language Models (MLLMs) have demonstrated a wide range of capabilities across many domains, including Embodied AI. In this work, we study how to best ground a MLLM into different embodiments and their associated action spaces, with the goal of leveraging the multimodal world knowledge of the MLLM. We first generalize a number of methods through a unified architecture and the lens of action space adaptors. For continuous actions, we show that a learned tokenization allows for sufficient modeling precision, yielding the best performance on downstream tasks. For discrete actions, we demonstrate that semantically aligning these actions with the native output token space of the MLLM leads to the strongest performance. We arrive at these lessons via a thorough study of seven action space adapters on five different environments, encompassing over 114 embodied tasks.

1 Introduction

Multimodal Large Language Models (MLLMs), defined as Large Foundation Models that take as input text and images and generate text, have recently seen rapid progress and impressive performance [1–13]. These models are important as they solve a large range of useful yet difficult natural language and image tasks, such as describing images, answering visual and textual questions, reasoning, and learning from a small number of examples. They have only recently improved to the point of being usable enough for general deployment with human non-experts [14–16].

While MLLMs are capable of describing real-world embodied concepts, their capabilities in embodied tasks are limited to using text for actions through generating code [17, 18], representing actions as text [19], or extracting actions from internal representations [20, 21]. *Grounding* [22] MLLMs to generate actions extends their capabilities to embodied tasks, such as robot manipulation and navigation, and is of tremendous value for practical problems, potentially overcoming the high cost of training tabula rasa. Extending MLLMs to multimodal image generation enables object detection and segmentation, and image and video generation [3, 23–27]. In embodied settings, grounding MLLMs via predicting agent affordances and generating actions yields effective policies capable of generalizing to new tasks [19, 21, 28, 29].

A key and open challenge in grounding MLLMs, which limits their capabilities in embodied tasks, is the gap between the native output space, natural language, and the action space of embodied agents. This problem is particularly acute in continuous action spaces, where low-level controllers may require a high degree of precision. Across the literature, a number of architectures and ways of handling action spaces have been proposed, but there has not been a systematic study of these designs. Our contributions generalize prior attempts to adapt MLLMs to generate actions through an empirical study on which principles and strategies are necessary to effectively close the gap between the action spaces of MLLMs and embodied agents. We study various grounding re-parameterization

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

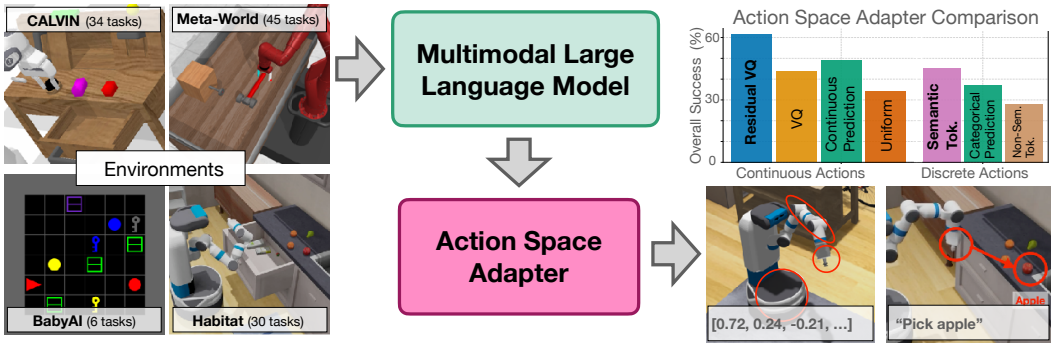


Figure 1: We empirically analyze how to ground MLLMs in actions across 114 tasks in continuous and discrete action spaces. In each environment, we train a multi-task policy with different Action Space Adapters (ASAs) to re-parameterize the MLLM to output actions. For continuous actions, learning a tokenization with several tokens per-action performs best (Residual VQ). For discrete actions, mapping actions to semantically related language tokens performs best (Semantic Tokenization).

strategies, which we refer to as Action Space Adapter (ASA), across a range of embodiments, action spaces, and environments. In particular, we explore the following types of ASAs: (1) ASAs that directly generate actions from a new prediction policy using the MLLM hidden representations as input; (2) ASAs that reuse the native token space of the MLLM to encode actions; (3) and ASAs that introduce a new token space to encode the actions of the agent while adapting the MLLMs to predict these new tokens.

Further, we empirically identify important principles for designing ASAs. For continuous action spaces, learned tokenization with several vocabularies that residually model continuous actions gives the right modeling precision while using vocabularies of manageable sizes and, as a result, yields the best performance across all continuous control environments. This learned tokenization outperforms direct action prediction, indicating this approach allows the model to effectively learn a multimodal distribution over action spaces. In addition, the above tokenization strategy boosts performance when the policy is a MLLM, compared to other standard non-LLM-based policies, indicating that it manages to better tap into the knowledge of the model.

For discrete action spaces, we study ASAs that better align the embodied actions with the output space of the MLLM. We demonstrate that a semantic alignment between these – mapping discrete actions to semantically related tokens in the MLLM vocabulary – yields the best strategy compared to other adapters that either reuse or define a new vocabulary. The superiority of this strategy is evident in performance on environments with discrete action spaces and also in RL sample efficiency.

Finally, the above principles are thoroughly validated across five embodied AI environments, three of which are robotic continuous control and two with discrete actions as illustrated in Figure 1. Altogether, we consider 114 language specified tasks. In the continuous case, the best tokenization achieves 72% on CALVIN [30], up from 68% for direct action regression and 28% for uniform action tokenization; and 84% on Meta-World [31], up from 61% for direct action regression and 75% for uniform tokenization. Similarly, in the case of discrete actions, the proposed semantically aligned action tokens yield 51% on LangR [21], up from 42% for direct action prediction.

2 Related Work

Prior works propose different Action Space Adapters (ASAs) to adapt MLLMs into policies. Some works use LLMs or MLLMs as zero-shot policies by prompting them to output text or code that can be executed as actions [18, 32–38]. The ASA in this case is a given executor or low-level controller that takes text as input and outputs actions in the environment. Other works investigate adapting MLLMs for actions, but focus on a single ASA and environment. For example, RT-2 [19] uniformly discretizes continuous actions and predicts tokens corresponding to each of the action dimensions. RoboFlamingo [20], Lamo [39], and LLaRP [21] use an MLP to predict an environment action from an LLM hidden state. GFlan [40] treats discrete actions as text and ranks actions by the LLM log

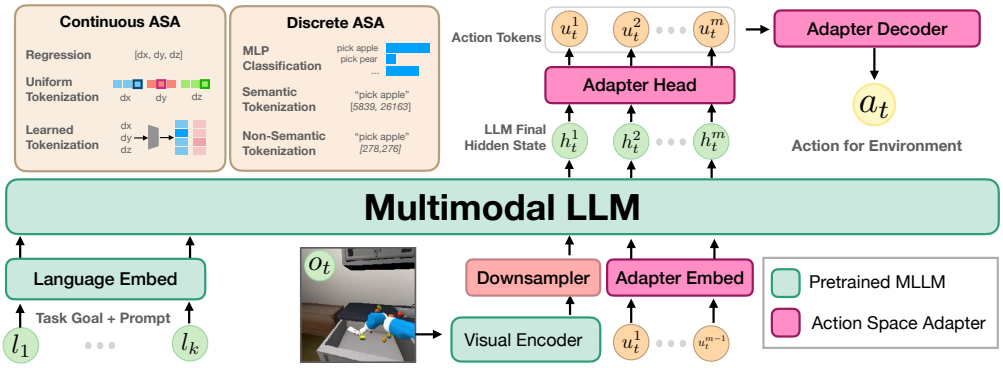


Figure 2: Generic architecture studied here for adapting MLLMs for action-specific decision making. The MLLM takes the embedding of the task instruction, prompt, and visual tokens as input. The MLLM then autoregressively predicts a sequence of m action tokens. These action tokens are then decoded into an environment-specific action.

probability to form a distribution over actions. At a high level, our work is distinct in that we study a variety of methods across multiple environments for learning ASAs. We focus on tasks with low zero-shot VLM performance, such as low-level control or long-horizon planning tasks. We summarize the differences between our investigation and prior work adapting VLMs for action in Appendix A.

Investigating action representations in embodied settings is not new. Some works learn representations of actions to help generalization to new actions or operating in large action spaces [41, 42] in the context of Reinforcement Learning (RL). Our study proposes ASAs for tokenizing continuous actions, and other works use different types of discretization or tokenization strategies on continuous action spaces. [43, 44] use k-means to discretize continuous actions to help learn from multimodal behavior datasets, such as from play data or data from different experts. VQ-BeT [45] finds learning a residual VQA (RVQ) codebook for continuous actions works best but does not apply this idea to MLLMs. [46] predicts actions as text. [47] learns a multi-task transformer policy and models actions with a diffusion head.

More broadly, prior works have adapted MLLMs for modalities other than actions, such as object bounding boxes and image generation, both being continuous in nature while the latter of high dimension. For example, [27, 48] train MLLMs to output spatial reference tokens to ground text responses in image regions. For image generation, [49] adapt MLLMs to generate image patches; [50, 51] tokenize images using a VQ-VAE model and adapt MLLMs to generate images by decoding these image tokens, which has inspired us to use the same learned tokenization; [52] uses an RVQ model [53] to generate images, similarly to our best performing tokenization scheme.

3 Method

In order to solve an embodied task, an agent learning in an interactive environment must select a decision from a set of valid actions. For example, an action space could be a set of keyboard presses for a video game or a real-valued vector that controls a robotic manipulator. Our work studies how to best adapt a MLLM, which is originally trained to output text tokens, to instead model actions from a given environment. We refer to the module that bridges a MLLM with a certain action space as an *Action Space Adapter* (ASA) (see Figure 2).

3.1 Problem Setting

Our analysis focuses on language-specified tasks with visual observations. Specifically, we consider a goal-specified Partially-Observable Markov Decision Process (POMDP) [54] that has an observation space $\mathcal{O}$, action space $\mathcal{A}$, and goal space $\mathcal{G}$. For brevity, we omit other elements of the MDP. In our setting, $\mathcal{G}$ is a textual description of the task to solve. $\mathcal{O}$ consists of RGB visual perception and agent proprioception. We consider a range of different action spaces $\mathcal{A}$ that broadly fall into two categories – discrete and continuous. The primary objective is to learn a language-conditioned policy that maps

observations and the instruction text to an action $\pi(a|o, g)$. As later described in Section 3.3, we learn this policy through supervised fine tuning from expert demonstrations or reinforcement learning that maximizes the expected discounted cumulative reward of the POMDP.

3.2 From Vision and Language to Action

The process studied here for adapting MLLMs for decision making is illustrated in Figure 2. The MLLM policy takes as input a textual instruction describing the downstream task, a sequence of past observations in the task and outputs an action in the agent's action space. In the bottom left of Fig. 2, the task description, as well as the environment description, are first encoded to produce language embeddings. To these embeddings, the MLLM then appends a sequence of visual embeddings from the current observation o_t. Since visual embeddings can often be comprised of a large number of tokens (the popular LLaVA-1.5 model [6] has 556), we introduce a downsampling layer to enable the MLLM to attend over a longer history of observations. In practice, we take the downsampling layer to be a Perceiver model [55], a learnable transformation that reduces the number of tokens from the visual encoder before being used as input to the MLLM.

The sequence of language and visual embeddings is passed through the MLLM, whose final hidden state h_t^1 encodes the entire input. The ASA, whose trainable parameters are denoted θ, is comprised of three parts: (1) an adapter head, (2) an adapter embedding, and (3) an adapter decoder. The hidden state is first passed through the adapter head to produce action tokens $u_t^1 = A_\theta(h_t^1)$. The action tokens are then embedded using the action embedding into $E_\theta(u_t^1)$, and passed autoregressively through the MLLM to produce further hidden embeddings $h_t^2, \ldots, u_t^m$ and associated action tokens $u_t^2, \ldots, u_t^m$, resulting in total m tokens per time step. The predicted action tokens are then decoded into the final action a_t by the adapter decoder, which produces the final action $a_t = D_\theta(u_t^1, .., u_t^m)$. As $a_t \in \mathcal{A}$, it is then executed in the environment to produce o_{t+1}, and the process continues.

Next, we describe possible ASA implementations for discrete and continuous action spaces.

3.2.1 Discrete Action Spaces

We define the following action spaces adapters for a discrete action space $\mathcal{A}$:

Categorical Prediction (Pred): Implement the action space adapter as an MLP network, which predicts the logits of a categorical distribution over environment actions from the MLLM hidden state. The adapter head is an MLP that maps the hidden state h^1 directly to an action $a \in \mathcal{A}$. This amounts to producing a single action token u^1, which directly corresponds to the action a, with the action decoder being an identity map. Both the adapter head and token embeddings are initialized from scratch. This type of ASA is used by [21].

Semantic Language (SemLang): The action space adapter predicts natural language text that maps to a discrete action. First, each action $a \in \mathcal{A}$ is described with freeform text tokenized as $(l_1, \ldots, l_m)$. The MLLM then autoregressively predicts a sequence of m tokens, which are then decoded by the adapter decoder to the corresponding action. For example, in an action space choosing a high-level skill a could be described as "pick apple", which is tokenized as $[5839, 26163]$ with the LLaMA tokenizer. The MLLM then must sequentially predict token 5839, then token 26163 to call this action. Sequences of tokens corresponding to invalid actions are either avoided entirely with the token filter described in Section 3.3 or treated as a no-op. Both the adapter head and the token embeddings are re-used to be the pretrained LLM's language head and embedding layer, respectively, meaning no additional parameters over the pretrained MLLM are added. This type of ASA is used by [29].

Non-Semantic Language (Lang): Actions are mapped to language tokens, but instead of semantically meaningful descriptions of the actions as with SemLang, the actions are mapped to sequences of numbers. For example, "pick apple" is represented with the string "5 3". The policy must then output the tokens corresponding to this text to call this pick action. Note that we can pick any text for this mapping and the choice of integers is arbitrary. However, the selected text is not semantically representative of the action.

3.2.2 Continuous Action Space Adaptors

We define the following four ASAs for a continuous D-dimensional action space $\mathcal{A}$: the first ASA predicts in the original action space while the other three use tokenization. At training time, we learn

a policy to predict these action tokens from the ASA. At test time, we employ an action decoder that maps these action tokens to actions in the original space $\mathcal{A}$.

Continuous Regression (Pred): Regress to the original continuous action from the MLLM hidden state h_t^1. This is achieved via a single-layer MLP network, which is trained using MSE loss. This ASA is used by [20, 39].

Uniform Action Tokenization (Uniform): The simplest approach is to use uniform binning of the action space. In particular, we express each action as a sequence of D tokens by quantizing each of the D action dimensions into one out of K uniform bins:

$$\text{Uniform(a)} = (k_1 \dots k_D) \quad \text{such that} \quad a_d \in \text{bin}(k_d, d)$$

where $\text{bin}(k, d)$ denotes the k^{th} bin along the d^{th} action dimension. If m_d and M_d denote the lower and upper bounds respectively of the d^{th} action dimension, then its definition reads $\text{bin}(k, d) = [m_d + k\frac{M_d - m_d}{K}, m_d + (k+1)\frac{M_d - m_d}{K}]$. At test time, we decode predicted action tokens to the center of the corresponding bins for each dimension. This type of ASA is used by [19].

Vector Quantized Tokenization (VQ): To adapt the tokenization to the particular action space, we propose to use learned tokenization. In particular, we express each action as a single token that corresponds to the closest action code from a learned codebook V. Using encoder network f_θ that maps actions to a latent embedding space:

$$\text{VQ}(a) = (k_1) \quad \text{where} \quad k_1 = \arg\min_k ||f_\theta(a) - v_k||_2^2$$

where $v_k \in V$. The codebook V of size K is learned over an offline dataset $\mathcal{D}$ of actions using a VQ-VAE [56] trained with the mean-squared error for action reconstruction and commitment loss. We overwrite K infrequently used tokens from the LLM vocabulary to represent V. We defer the full details of this tokenization process to Appendix C.2.

Residual Vector Quantized Tokenization (RVQ): Precise control requires precise action modeling that can suffer after tokenization. To increase the precision of a learned tokenization, we further investigate the use of a sequence of several action tokens as in Uniform. Similar to VQ, these tokens are from M action codebooks $V_m, m \in \{1, \dots, M\}$. However, each codebook models the residual space obtained after modeling the action using preceding codebooks, thus each subsequent token captures increasingly finer action information:

$$\text{RVQ}(a) = (k_1, \dots k_M) \quad \text{where} \quad k_m = \arg\min_k \left\| \left(f_\theta(a) - \sum_{i=1}^{m-1} v_{k_i}^i \right) - v_k^m \right\|_2^2$$

where $v_k^i \in V_i$ is the k^{th} code from the i^{th} codebook. Such tokenization can be learned using Residual VQ-VAE [RVQ-VAE, 52] on an offline dataset of actions. The actual number of token sequences we can represent is K^M. Hence, RVQ presents the opportunity to exponentially increase the action space quantization without having to drastically increase the size of the learned individual codebooks.

3.3 Training

We use LLaVA-1.5-7B [6] as the base MLLM. We finetune the MLLM with interactive (i.e., action-labeled) data to make it more suited for interacting with a embodied and interactive environment.

Supervised Fine Tuning (SFT) with Expert Demonstrations: We finetune the MLLM for interactive tasks using a dataset of expert demonstrations. Each demonstration contains (1) a language description of the task, (2) a sequence of observations, and (3) a sequence of actions that successfully solve the task. Note that in this work, we are primarily interested in learning imitation policies from offline data, which can be extended to offline reinforcement learning if per-timestep rewards are included in the dataset. Specifically, we train the MLLM with supervised learning to predict the expert actions from the observations and language description in the data. While the pre-trained LLM and the visual encoder remain frozen, we finetune the ASA, the visual downsampler, and parts of the LLM with LoRA [57]. In total, the model has $\approx 100M$ learnable LLM parameters and $\approx 40M$ learnable downsampler and ASA parameters. The learned tokenization schemes (RVQ and VQ) have an additional pre-training phase, where the VAE models are first trained on actions from the offline dataset and then frozen to prevent further updates in later stages.

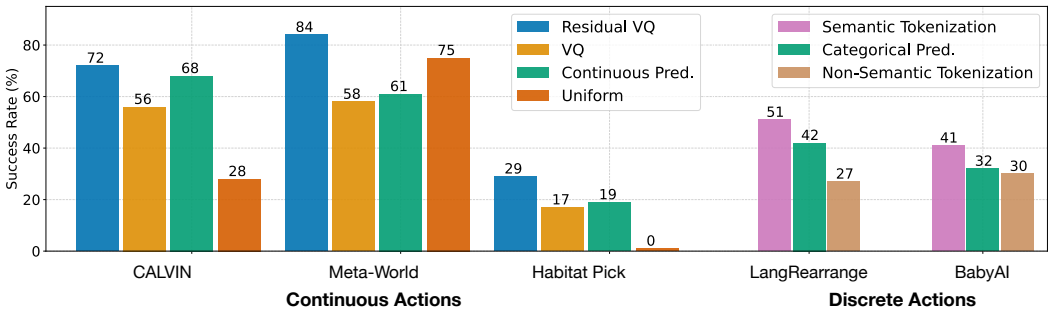


Figure 3: Comparing ASAs for continuous and discrete action spaces across 5 environments. For continuous actions, the RVQ tokenization performs best. For discrete actions, SemLang performs best. Each bar gives the average over all tasks in the environment with the full breakdown in Appendix E.

Reinforcement Learning (RL) from Environment Feedback We can also optionally finetune the MLLM to optimize an environment reward using RL. However, predicting actions in the MLLM token space dramatically increases the number of possible action predictions, with many possible predictions corresponding to no valid action. For example, there are 32,000 tokens in the LLaMA text tokenizer, giving $32,000^m$ possible predictions by the model with m tokens per action. This makes exploration difficult in RL as only a small fraction of the possible actions are valid. We therefore use a *token filter* to restrict the autoregressive sampling to only be from token sequences corresponding to valid actions. The token filter is a function $M(l_t^1, \ldots, l_t^{j-1})$ that produces a binary mask over all tokens to represent valid tokens for the jth decoding step.

4 Experiments

4.1 Experimental Settings

We study adapting MLLMs for action across a variety of environments with different embodiments and action spaces. All environments provide RGB visual observations and a natural language instruction specifying the goal to achieve. We provide the important environment details below and defer complete details to Appendix B.

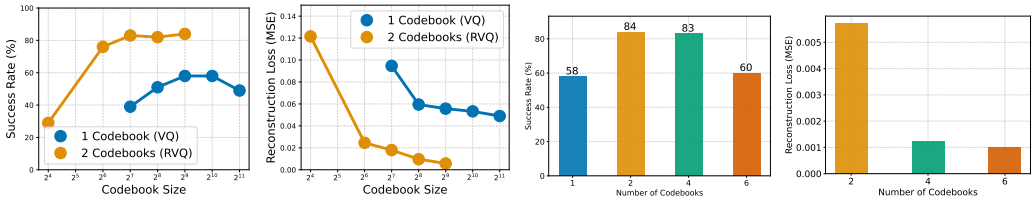
CALVIN [30]: This manipulation benchmark tests the ability of a tabletop robot to interact with an object to complete a natural language instruction. The continuous actions specify 6DoF end-effector control and the binary gripper state. The observation is a 200×200 RGB image from a fixed-position camera. We use the $ABC \rightarrow D$ split of the benchmark with 34 tasks, and the agent is evaluated on unseen instruction phrasings and table background.

Meta-World [58]: We use the ML-45 version of this tabletop manipulation benchmark which has 45 tasks. The action space is continuous control specifying 3DoF end-effector translation and the continuous gripper state. The observations are 200×200 RGB images from a fixed camera. The agent is evaluated on unseen object and robot starting states.

Habitat Pick (HabPick) [59]: A mobile manipulation robot must pick up an object specified by name from a receptacle. The continuous actions specify the 7DoF relative joint positions of the arm, the 2D base velocity, and the gripper state. The observations are 336×336 RGB images from the robot's egocentric head camera. The instruction specifies the name of the object type to pick up. The evaluation distribution is on unseen houses and new arrangements of objects.

BabyAI [60]: BabyAI is a grid world task where an agent navigates and interacts with objects to complete an instruction. The discrete action space consists of navigation and interaction actions. The observation is a 200×200 RGB top-down view. We use the five tasks from [40], and we report generalization to instructions rephrased with synonyms.

Language Rearrangement (LangR) [21]: A mobile manipulation robot must rearrange objects to complete instructions like "store all the fruit in the fridge". The discrete actions are 70 high-level skills to interact with objects and navigate. The observation is a 336×336 RGB head camera.



(a) # Codes: Success (b) # Codes: Recon. (c) # Codebooks: Success (d) # Codes: Recon.

Figure 4: (a,b) show the effect of the number of codes in the codebook for RVQ and VQ on final policy success rate (see (a)) and reconstruction on unseen action trajectories in Meta-World (see (b)). (c,d) show the effect of number of codebooks on final policy success rate (see (c)) and action reconstruction (see (d)). All metrics are computed on Meta-World.

Evaluation instructions test generalization to unseen houses and 10 unseen instruction datasets measuring paraphrastic robustness and behavior generalization.

In all environments, we report the success rate as the fraction of episodes in which the agent completed the language instruction. We use the success criteria provided by each environment. We train a policy per action adapter for each environment and report the generalization performance in the main text. When reporting a single success rate per environment, it is the success averaged between all evaluation episodes containing all tasks. We give the full per-task breakdown for results in Appendix E. CALVIN, Meta-World, HabPick, and BabyAI provide expert demonstrations succeeding at the task. CALVIN has $17.9k$ from humans, Meta-World $22.5k$ from a scripted policy, HabPick $6.7k$ generated from an RL policy, and BabyAI $5k$ from a scripted policy. Full details on the train and evaluation setups per environment are in Appendix B.

We train with supervised finetuning for CALVIN, Meta-World, HabPick, and BabyAI. We train with reinforcement learning on Language Rearrangement. As described in Section 3.3 we train $\approx 140M$ parameters with LoRA [57]. We use the AdamW optimizer [61] with a learning rate of 3e^{-4}, a warmup period of 10% of the total number of training steps, and cosine learning rate decay to 0 by the end of training. For RL, we use PPO [62]. For the learned tokenization action space adapters, we, by default, use a codebook size of 512 with 512 dimensions per codebook element. Complete hyperparameter and policy details are in Appendix C.

4.2 Continuous Action Space Adapter Comparison

We first study adapting MLLMs through Uniform, Pred, VQ, and RVQ action space adapters for the continuous action environments CALVIN, Meta-World and HabPick.

RVQ is the best performing continuous action ASA. The results in Figure 3 show that the RVQ action adapter consistently outperforms all other ASA approaches across all environments. While Pred is the second best performing method on all tasks, except on Meta-World, RVQ outperforms it by a 12% average absolute difference. One hypothesized reason for this is that Pred only learns unimodal distributions of actions, which hurts performance when learning from diverse demonstrations [43–45]. Another potential reason is the tokenization from RVQ allows the MLLM to better leverage its existing knowledge, whereas the Pred ASA requires training a new MLP network from scratch.

Uniform performs poorly on the majority of the tasks, where RVQ outperforms on average by a 27% absolute increase. A reason for this is that the Uniform discretization can fail to accurately represent the continuous actions. The performance of Uniform is also closely related to the action dimension. In Meta-World with 4 action dimensions, Uniform performs well. However, Uniform suffers with the 7 action dimensions in CALVIN and the 10 action dimensions in HabPick.

RVQ also outperforms VQ by a 18% absolute difference averaged over all environments. This is due to VQ having worse action reconstructions than RVQ. In Meta-World, both RVQ and VQ policies reach a similar cross-entropy loss on holdout trajectories during finetuning. However, on this same data, RVQ has a reconstruction mean squared error (MSE) of 0.005 while VQ has a 10x higher reconstruction MSE of 0.05. Increasing the VQ codebook size does not close this gap. We vary the VQ codebook size in powers of 2 from 2^7 to 2^{11}. Figure 4b shows the VQ reconstruction loss decreases with larger codebooks but does not even close the gap to the 2^7 RVQ codebook size. This poor reconstruction manifests in poor downstream policy performance as demonstrated by Figure 4a

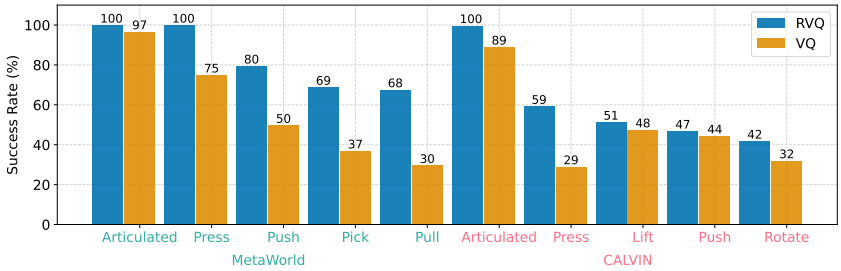


Figure 5: RVQ and VQ success per-task grouping (defined in Supp. B.6) on CALVIN and MetaWorld.

where policies trained with the VQ ASA plateau in success rate at codebook size 2^9. VQ policies even decrease in performance at codebook size 2^{11}, potentially due to overfitting to the large codebook.

We further characterize the performance of RVQ and VQ in Figure 5 by breaking down the performance per task group in Meta-World and CALVIN. The task groups, which are fully listed in Appendix B.6, correspond to tasks with related required behaviors. Both RVQ and VQ do similarly on "articulated" object interactions (like opening drawers or doors). These tasks require less precise control since many contact points on the articulated link and broad pushing or pulling behavior can achieve the desired behavior. On the other hand, RVQ outperforms VQ on "pressing" tasks that require pushing a button. These tasks require more precise control since the agent needs to push the button all the way to a desired state. VQ often reaches the button but fails to press it all the way. The same is also true of other precise control tasks like picking, pulling, and rotating.

A potential explanation of RVQ's success can be attributed to adaptive localization of the model's errors, similar to prior work in residual reinforcement learning [63] and Bellman error bases [64].

A sufficient codebook size and number of codebooks are necessary for RVQ. In Figure 4a, we show that RVQ policy performance improves in performance with a larger codebook size in Meta-World. Notably, RVQ performs poorly at 29% success rate with codebook size 16 compared to 84% success at codebook size 512. These observations also align with the codebook size decreasing reconstruction error in Figure 4b. In Figure 4c, we compare the effect of the number of codebooks on performance. As earlier discussed with the performance of VQ, one codebook results in poor action reconstruction and, thus, bad policy performance. However, increasing the number of codebooks too much to 6 also hurts performance despite decreasing reconstruction loss. Likewise to the finding that Uniform performs poorly with larger action dimension since there are more tokens per action, increasing the number of codebooks also hurts policy learning.

RVQ tokens transfer to new tasks. We take the model trained on the 45 Meta-World tasks and finetune it on 5 unseen tasks. We collect 50 demonstrations for per task and finetune the policy on all task data. We use the same RVQ ASA trained only on data from the 45 tasks. Figure 6a shows the success rate of adapting RVQ compared to an Pred ASA. RVQ outperforms Pred across all tasks, achieving a 50% vs. 20% overall success rate. This demonstrates the RVQ tokens are flexible enough to be applied to new tasks.

The gains from RVQ are unique to MLLMs. Next, we analyze the unique interaction between the RVQ tokens and the MLLM policy. While we demonstrated that the RVQ ASA performs best, is this improvement due to the MLLM being able to leverage these new tokens or the added action representation ability from the separately trained RVQ decoder? To test this, we compare to two policy architectures that do not use LLMs:

- **Scratch**: This is the same architecture as the MLLM-based policy, but with a smaller 300M parameter non-pretrained transformer.
- **RT-Inspired**: This method uses a ResNet visual encoder, pretrained Flan [65] language embedding and decoder transformer-based policy. The entire policy is trained from scratch. This method is inspired by RT-1 [66], which does not have publicly released code.

Table 1 compares the effect of Pred versus RVQ ASAs on CALVIN, Meta-World and HabPick for these three policy policy architectures. As already established for the MLLM, RVQ is consistently better than VQ. However, for the same policy architecture trained from scratch, RVQ can hurt the performance over Pred. In CALVIN the success drops -7% and in Meta-World the performance

	MLLM: Pred→RVQ	Scratch: Pred→RVQ	RT-Inspired: Pred→RVQ
Calvin	68 → 72 (+4)	50 → 43, (-7)	35 → 36, (+1)
Metaworld	61 → 84 (+23)	71 → 56, (-15)	27 → 38, (+11)
Habitat Pick	19 → 29 (+10)	21 → 25, (+4)	18 → 20, (+2)

Table 1: Comparing the effect of the RVQ action space adapter on the success rate of non-LLM based policies. Red indicates RVQ hurts over Pred and green indicates RVQ helps over Pred. RVQ typically has a negative impact on the Scratch policy, and helps the smaller RT-Inspired policy.

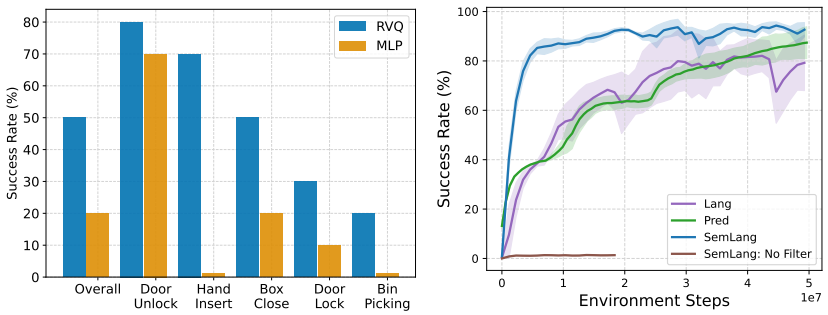
drops -15%. This highlights that MLLM can leverage its existing knowledge about sequencing language tokens to sequencing action tokens. However, we find that for the smaller RT-Inspired policy network, the RVQ ASA consistently helps, which we hypothesize is because the added RVQ network and separate training help compensate for the lack of policy network capacity. We also note that RVQ may more consistently outperform Pred on demonstrations that explicitly contain multimodal action sequences [43–45].

4.3 Discrete Action Adapter Comparison

SemLang performs the best. In Figure 3, SemLang outperforms the next best ASA (Pred), by 9% on Language Rearrangement and 8% on BabyAI. SemLang performs especially well on tasks with explicit high-level language actions in Language Rearrangement (e.g., "pick apple") where prior work has shown text-only LLM policies achieve non-zero success [21]. SemLang also does well on the BabyAI tasks with discrete low-level actions like "move left". Additionally, Lang performs the worst in both environments, achieving 14% lower success on Language Rearrangement and 11% lower on BabyAI than SemLang. We hypothesize this is because the MLLM has to repurpose its knowledge to leverage these newly assigned action tokens, whereas a newly initialized Pred allows extracting this knowledge from the MLLM hidden state.

SemLang enables sample efficient RL. In Figure 6b, we compare the RL training curves for the ASAs in Language Rearrangement. In addition to helping with better generalization, SemLang also enables sample efficient RL training. SemLang converges in training performance after just $20M$ training samples, whereas Pred requires up to $70M$ steps to fully converge.

Token filter is crucial for language-based action spaces. In Figure 6b, we show the training of SemLang without the token filter, which restricts policy outputs to only valid action token sequences. Without the token filter, SemLang is unable to learn in the large text action space.



(a) Finetuning on Holdout Tasks (b) Language Rearrangement RL.

Figure 6: (a) Adapting to 5 holdout tasks from Meta-World ML-45 with 50 demos per task using the fixed RVQ tokenization. (b) RL training curves in Language Rearrangement comparing the ASAs and utility of the token filter. Displayed are averages over 2 seeds with the shaded area as the standard deviation between seeds. SemLang learns faster than other ASAs and the token filter is crucial.

4.4 Empirical Comparison to Prior Work

The contributions of this work are an empirical analysis of ASAs under controlled settings on various embodied environments. Direct comparisons to prior work are challenging due to different training algorithms, policy architectures, or assumptions about input modalities. Regardless, in this section, we seek to contextualize our RVQ and SemLang MLLM results against prior work. In Meta-World, to the best of our knowledge, RVQ at 84% success on ML-45 sets a new state-of-the-art result, compared to 79% from DualMind [67]. In CALVIN, RVQ at 72% success underperforms a similar work RoboFlamingo which achieves 82% success on the $ABC \rightarrow D$ split. However, RoboFlamingo uses a different MLLM and uses an additional gripper camera input. In Language Rearrangement, SemLang sets a state-of-the-art result with 51% success compared to 42% from LLaRP [21]. In BabyAI, SemLang at 40% success rate underperforms GFlan [40], which achieves 55% success. However, we use RGB visual observations, while GFlan operates from a compact, ground truth language state description. In Appendix A.1, we compare these differences in more detail.

5 Limitations and Conclusion

In this work, we studied various action space adapters (ASAs) across a variety of embodiments, action spaces, and environments. We provide a generalization of prior works through the lens of action space adaptors, and for both discrete and continuous action spaces demonstrate designs that we show can leverage the knowledge within the MLLM. Our findings conclude that for continuous actions, it is best to learn action tokens that accurately model the action distribution, while for discrete actions, it is best to reason over semantic language descriptions of actions. We verify these ideas across 114 embodied AI tasks in 5 diverse environments.

A limitation of our work is all our analysis is under a single MLLM (LLaVA). Another limitation is that RVQ, the best performing ASA in continuous action spaces, requires collecting demonstrations to train the VQ model. Our analyses are also under only a single LoRA training setting. Future analyses can explore different base MLLMs under different training regimes like full LLM finetuning. While our investigation of ASAs enables connecting a MLLM to various action spaces, the performance of these methods is still subpar for real-robot deployment where high success and safety are critical. MLLMs with the best ASA still struggle on simple environments like BabyAI, only achieving 40% success rate. Further work is needed to improve the performance of these methods for real-world usage. Our investigation also only studies adapting MLLMs through behavioral cloning or on-policy RL. Future work can investigate if the choice of ASA varies when adapting the MLLM with other learning algorithms such as off-policy RL or offline RL.

References

[1] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

[2] Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, Qiang Liu, Kriti Aggarwal, Zewen Chi, Johan Bjorck, Vishrav Chaudhary, Subhojit Som, Xia Song, and Furu Wei. Language is not all you need: Aligning perception with language models, 2023.

[3] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023.

[4] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.

[5] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023.

[6] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[7] Chunyuan Li, Zhe Gan, Zhengyuan Yang, Jianwei Yang, Linjie Li, Lijuan Wang, and Jianfeng Gao. Multimodal foundation models: From specialists to general-purpose assistants. *arXiv preprint arXiv:2309.10020*, 2023.

[8] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[9] Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*, 2023.

[10] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023.

[11] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Mimic-it: Multi-modal in-context instruction tuning. *arXiv preprint arXiv:2306.05425*, 2023.

[12] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*, 2024.

[13] IDEFICS. Introducing idefics: An open reproduction of state-of-the-art visual language model. https://huggingface.co/blog/idefics, 2023.

[14] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[15] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[16] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[17] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *arXiv preprint arXiv:2209.07753*, 2022.

[18] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[19] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[20] Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. *arXiv preprint arXiv:2311.01378*, 2023.

[21] Andrew Szot, Max Schwarzer, Harsh Agrawal, Bogdan Mazoure, Walter Talbott, Katherine Metcalf, Natalie Mackraz, Devon Hjelm, and Alexander Toshev. Large language models as generalizable policies for embodied tasks. *arXiv preprint arXiv:2310.17722*, 2023.

[22] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55, 2020.

[23] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic. *arXiv preprint arXiv:2306.15195*, 2023.

[24] Haoxuan You, Haotian Zhang, Zhe Gan, Xianzhi Du, Bowen Zhang, Zirui Wang, Liangliang Cao, Shih-Fu Chang, and Yinfei Yang. Ferret: Refer and ground anything anywhere at any granularity. In *ICLR*, 2024.

[25] Wenhai Wang, Zhe Chen, Xiaokang Chen, Jiannan Wu, Xizhou Zhu, Gang Zeng, Ping Luo, Tong Lu, Jie Zhou, Yu Qiao, et al. Visionllm: Large language model is also an open-ended decoder for vision-centric tasks. *arXiv preprint arXiv:2305.11175*, 2023.

[26] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023.

[27] Hao Zhang, Hongyang Li, Feng Li, Tianhe Ren, Xueyan Zou, Shilong Liu, Shijia Huang, Jianfeng Gao, Lei Zhang, Chunyuan Li, et al. Llava-grounding: Grounded visual chat with large multimodal models. *arXiv preprint arXiv:2312.02949*, 2023.

[28] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

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

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

[31] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[32] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on Robot Learning*, pages 492–504. PMLR, 2023.

[33] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[34] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500. IEEE, 2023.

[35] Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, et al. Grounded decoding: Guiding text generation with grounded models for robot control. *arXiv preprint arXiv:2303.00855*, 2023.

[36] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: Personalized robot assistance with large language models. *arXiv preprint arXiv:2305.05658*, 2023.

[37] Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Pack Kaelbling, and Michael Katz. Generalized planning in pddl domains with pretrained large language models. *arXiv preprint arXiv:2305.11014*, 2023.

[38] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[39] Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon S Du, and Huazhe Xu. Unleashing the power of pre-trained language models for offline reinforcement learning. *arXiv preprint arXiv:2310.20587*, 2023.

[40] Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. *arXiv preprint arXiv:2302.02662*, 2023.

[41] Ayush Jain, Andrew Szot, and Joseph J Lim. Generalization to new actions in reinforcement learning. *arXiv preprint arXiv:2011.01928*, 2020.

[42] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[43] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[44] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[45] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[46] Georgios Pantazopoulos, Malvina Nikandrou, Amit Parekh, Bhathiya Hemanthage, Arash Eshghi, Ioannis Konstas, Verena Rieser, Oliver Lemon, and Alessandro Suglia. Multitask multimodal prompted training for interactive embodied task completion. *arXiv preprint arXiv:2311.04067*, 2023.

[47] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[48] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023.

[49] Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Zhengxiong Luo, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, et al. Generative multimodal models are in-context learners. *arXiv preprint arXiv:2312.13286*, 2023.

[50] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. Scaling robot learning with semantically imagined experience. *arXiv preprint arXiv:2302.11550*, 2023.

[51] Emanuele Aiello, Lili Yu, Yixin Nie, Armen Aghajanyan, and Barlas Oguz. Jointly training large autoregressive multimodal models. *arXiv preprint arXiv:2309.15564*, 2023.

[52] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022.

[53] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

[54] Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518.

[55] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[56] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

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

[58] Tianhe Yu, Deirdre Quillen, Zhanpeng He, R. Julian, Karol Hausman, Chelsea Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019.

[59] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34, 2021.

[60] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: First steps towards grounded language learning with a human in the loop. In *ICLR*, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

[61] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[62] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[63] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[64] Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 752–759, 2008.

[65] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[66] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[67] Yao Wei, Yanchao Sun, Ruijie Zheng, Sai Vemprala, Rogerio Bonatti, Shuhang Chen, Ratnesh Madaan, Zhongjie Ba, Ashish Kapoor, and Shuang Ma. Is imitation all you need? generalized decision-making with dual-phase training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16221–16231, 2023.

[68] Ankesh Anand, Jacob Walker, Yazhe Li, Eszter Vértes, Julian Schrittwieser, Sherjil Ozair, Théophane Weber, and Jessica B Hamrick. Procedural generalization by planning with self-supervised world models. *arXiv preprint arXiv:2111.01587*, 2021.

[69] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[70] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

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

[72] Andrew Szot, Karmesh Yadav, Alex Clegg, Vincent-Pierre Berges, Aaron Gokaslan, Angel Chang, Manolis Savva, Zsolt Kira, and Dhruv Batra. Habitat rearrangement challenge 2022. https://aihabitat.org/challenge/2022_rearrange, 2022.

[73] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[74] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[75] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

Work	Environments	Best ASA	Other ASAs Studied	Action Space Types	Training	Using LLM/VLM?
RoboFlamingo [20]	CALVIN	MLP	-	Continuous	BC	✓
Lamo [39]	Franka Kitchen, Atari, MuJoCo	MLP	-	Continuous, Discrete	Offline-RL	✓
GFlan [40]	BabyAI	Sem Lang (scoring)	-	Discrete	Online RL	✓
RT-2 [19]	Internal	Uniform	-	Continuous	BC	✓
LLaRP [21]	Language Rearrangement	MLP	-	Discrete	Online-RL	✓
VQ-BeT [45]	PushT, Multimodal Ant, BlockPush, Franka Kitchen, nuScenes, PlayKitchen	RVQ+MLP	-	Continuous	BC	✗
Ours	Language Rearrangement, Baby AI, Meta-World, CALVIN, Habitat Skills	RVQ/ Sem-Lang	MLP, VQ, Uniform, Non-Sem, Non-Sem Comp	Continuous, Discrete	Online-RL, BC	✓

Table 2: Comparing our investigation to prior work. Prior work typically analyzes a single action adapter in a single environment. We study a variety of action adapters across a variety of environments.

A Prior Work Comparison

In this section we expand on the differences between the prior work in action space adaptation mentioned in Section 2 and our investigation. Table 2 compares our investigation to prior work along several key dimensions. We emphasize that unlike prior works, ours studies a variety of action space adapters under a greater diversity of environments.

A.1 Empirical Comparison to Prior Work

We report performance on standard benchmarks which prior work has also extensively studied. However, even within the benchmarks there are differences in training algorithms and sensor input assumptions that make direct comparison to prior work difficult. Regardless of these differences, we study different ASAs for MLLMs in a consistent experimental setting. We also describe differences between the empirical setups of ours and prior works that perform well on these benchmarks.

Meta-World (MLLM +RVQ 84% success rate on ML-45): To the best of our knowledge, our 84% is the highest reported on Meta-World ML-45 so far. Anand et al. [68] operates under similar sensor assumptions and achieves 77% success with MuZero [69]. DualMind [67] achieves 79% success rate on ML-45 and outperforms other generalist agents like Gato [70]. However, DualMind uses privileged simulator information about the joint states and object positions while we only use RGB visual observations.

CALVIN (MLLM +RVQ 72% success rate): RoboFlamingo achieves a higher 82% success rate on the same $ABC \rightarrow D$ task. However, RoboFlamingo uses the OpenFlamingo VLM while we use LLaVA. RoboFlamingo use the gripper and fixed camera while we only use the fixed camera. More recent work like 3D Diffuser Actor [71] practically solves the $ABC \rightarrow D$ task, achieving 96% success rate. However, this work uses depth inputs, and a diffusion model policy that predicts keypoints for the end-effector rather than underlying actions. Our work uses only RGB visuals, uses a MLLM policy and predicts relative end-effector poses rather than keypoints.

Language Rearrangement (SemLang 51% success rate): This outperforms the prior highest reported number of 42% on the overall evaluation set from LLaRP [21].

BabyAI (SemLang 40% success rate): GFlan [40] achieves 55% success on the same evaluation split. However, the GFlan policy takes as input a ground truth language description of the state, while our policies take as input a 200×200 RGB top down rendering of the environment. GFlan also trains the policy with reinforcement learning while we train with supervised learning.

B Environment Details

An overview of the environments is visualized in Figure 7. This figure visualizes the training observations input to the agent. We run experiments on 5 environments, and each environment in turn consists of multiple tasks. We arrive at the task count of 114 in the main paper through 45 tasks in Meta-World, 34 in CALVIN, 20 in HabPick where we count each object goal as a different task, 10 in Language Rearrangement for each of the evaluation splits, and 5 in BabyAI. The task count for Language Rearrangement is conservative since technically it consists of 282 instruction templates, each of which corresponds to a distinct task and goal.

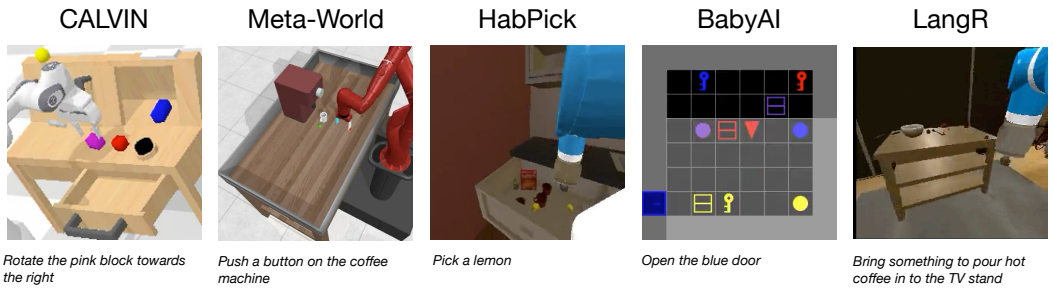


Figure 7: Visualizations of the environments we study. The top row shows an observation in the environment. The bottom row shows the associated instruction in that episode.

B.1 Meta-World

Tasks: We use the ML-45 benchmark from Meta-World [58]. Each of the 45 tasks are specified with a fixed language instruction. We use the task descriptions from Appendix Section A of Yu et al. [58].

Observation Space: 200×200 RGB images from a fixed camera position. To render the visual observations, we only use the "corner4" camera position as this gives an unobstructed view of the robot in most of the tasks.

Action Space: 4DoF continuous control of the arm and gripper. The first 3 dimensions specify the relative end-effector translation. The last dimension specifies the desired gripper state.

Training: We use 40 start and goal configurations for each of the tasks. We generate 500 demonstrations for each of the 45 tasks. We use the scripted policies from Yu et al. [58]. At each step we add Gaussian noise $\mathcal{N}(0, 0.1)$ to the actions produced by the scripted policy before executing it in the environment. We generate 500 successful trajectories per task, resulting in $45 \cdot 500 = 22.5k$ total trajectories.

Evaluation: We evaluate performance on 10 unseen start and goal configurations for each of the 45 tasks. So in total, we evaluate on 450 unseen configurations and report the average performance over these 450 episodes.

B.2 CALVIN

Tasks: We use the CALVIN $ABC \rightarrow D$ dataset split.

Observation Space: 200×200 RGB observations from the fixed camera view.

Action Space: 7DoF continuous control of the arm and gripper. The first 6 dimensions specify the relative position and rotation of the end-effector. The final dimension is a binary indicator for if the gripper should be open or closed. We hold out 1024 subsequences of the policy context length from these trajectories for reporting validation performance during the SFT process.

Training: We use the 17,871 demonstrations provided in the CALVIN $ABC \rightarrow D$ dataset. These demonstrations are in 3 different table backgrounds. This also includes 1,088 demonstrations for validation.

Evaluation: We report performance on the D split. This evaluation scene is a different color than that encountered during training. All the start positions and goals are also different. Many of the language instructions are also unseen from training. We report the average performance over the 1,000 evaluation sequences. We report the success of the first task completed in the sequence.

B.3 Habitat Pick

Tasks: We use the same Pick task as in Habitat 2.0 Geometric Goal object rearrangement [59, 72], except we provide the agent the name of the object to rearrange rather than the starting coordinates of the object and increase the observation resolution. The task is successful when the agent picks up the object and returns the end-effector within a fixed offset to a "resting position" in front of the robot.

The task ends in failure if the agent excessively collides with the scene, drops the object, or picks up the wrong object. The agent starts within 2 meters of the object and facing towards the receptacle but with random noise $\mathcal{N}(0, 1.57)$ applied to the direction of facing directly at the receptacle. The maximum number of steps per episode is 300 steps.

Observation Space: A 336×336 head-mounted RGB camera.

Action Space: The action space is 10DoF control of the arm, base and gripper. The first 2 dimensions control the linear and angular velocity of the base. The next 7 dimensions control the relative joint offsets of the arm. The final dimension controls whether the suction gripper is engaged or not.

Training: We first train a privileged policy with RL to complete the task. This policy takes as input the egocentric depth image and the ground truth position of the target object to pick up. We collect $20k$ successful trajectories.

Evaluation: We evaluate on the test episodes from Szot et al. [72] which are $1,000$ episodes in unseen home layouts.

B.4 BabyAI

Tasks: The tasks all occur in a 6×6 grid populated with interactable objects. We use the task definitions from Carta et al. [40]. This consists of the following 5 instruction templates: "Go to <object>", "Pick up <object>", "Put <object A> next to <object B>,", "Pick up <object A> then go to <object B> and Go to <object B> after pick up <object A>", "Unlock <door>". The maximum number of steps per episode is 50 steps.

Observation Space: 200×200 RGB observation as a top down of the 6×6 scene. Note this is a more challenging observation space than prior gridworld navigation tasks that provide the current view as a compact entity specific array [60] or by a language description [40].

Action Space: The action space consists of 6 actions consisting of: turn left, turn right, move forward, pick, drop and toggle.

Training: We collect 1,000 demonstrations for each of the 5 templates. We randomly sample an instruction and starting state configuration for every demonstration. We use the expert planner from Chevalier-Boisvert et al. [60] to generate the demonstrations.

Evaluation: We report performance on the unseen synonyms generalization test, described in Section 4.2 of Carta et al. [40]. We evaluate on 200 episodes per template type, giving 1000 total evaluation episodes.

B.5 Language Rearrangement

Tasks: An agent starts in an unseen house and must complete a rearrangement task from a language instruction.

Observation Space: The agent has a 336×336 head-mounted RGB camera. We increase the camera resolution from 256×256 in the original Language Rearrangement task to match the input resolution of the LLaVA CLIP encoder.

Action Space: We use the same action space as from the original Language Rearrangement benchmark Szot et al. [21]. The agent can select between 70 high-level skills that include picking up objects by name, navigating to receptacles, placing on receptacles by name, and opening and closing receptacles by name.

Training: Since Language Rearrangement does not provide any demonstrations and due to the emphasis on exploration in the problem, they are not readily obtainable, even with oracle planners. Therefore, we opt to train policies with reinforcement learning from the environment reward provided by the Language Rearrangement task.

Evaluation: We evaluate on all 10 evaluation datasets from Language Rearrangement consisting of 1,000 evaluation episodes on unseen scenes.

B.6 Task Groupings

In Section 4 we breakdown the performance on CALVIN and MetaWorld for task groupings. Each of the task groupings consists of multiple tasks from the benchmark. We grouped tasks in the following way:

MetaWorld:

- Articulated: “door-close", “door-open", “drawer-close", “drawer-open", “faucet-open", “faucet-close", “handle-press-side", “handle-press", “window-open", “window-close"
- Press: “button-press-topdown", “button-press-topdown-wall", “button-press", “button-press-wall", “coffee-button"
- Push: “plate-slide", “plate-slide-side", “plate-slide-back", “plate-slide-back-side", “push-back", “push", “push-wall", “stick-push", “sweep-into", “sweep", “soccer", “coffee-push"
- Pick: “assembly", “basketball", “dial-turn", “disassemble", “hammer", “peg-insert-side", “peg-unplug-side", “pick-out-of-hole", “pick-place", “pick-place-wall", “reach", “reach-wall", “shelf-place"
- Pull: “coffee-pull", “handle-pull-side", “handle-pull", “lever-pull", “stick-pull"

CALVIN:

- Articulated: “move slider left", “open drawer", “close drawer", “move slider right"
- Press: “turn off led", “turn on led", “turn on lightbulb", “turn off lightbulb"
- Lift: “lift blue block slider", “lift pink block table", “lift red block slider", “lift red block table", “lift pink block slider", “lift blue block table"
- Push: “push pink block right", “push blue block right", “push red block left", “push pink block left", “push red block right", “push blue block left", “push into drawer"
- Rotate: “rotate red block right", “rotate red block left", “rotate pink block left", “rotate pink block right", “rotate blue block right", “rotate blue block left"

C Further Policy Details

C.1 Prompt Details

In addition to inputting the task instruction to the LLM, we also format the instruction with a prompt. We base our prompt off the prompt used in LLaVA. For all continuous control tasks, we use the prompt template “Prompt: control the robot. USER: <INSTRUCTION> ASSISTANT: ". For discrete action space tasks, we describe the available actions to the agent in the prompt as well. For BabyAI, this is the prompt template “Prompt: Control the red triangle to complete the instruction using left, right, forward, pick, drop and toggle. USER: <INSTRUCTION> ASSISTANT: ". For Language Rearrangement, this is the prompt template “Prompt: You are a home robot assistant. Your possible actions are: pick object, place receptacle, nav receptacle, open receptacle, close receptacle, STOP. - Objects: ball, clamp, hammer, screwdriver, padlock, scissors, block, drill, spatula, knife, spoon, plate, sponge, cleanser, plum, pear, peach, apple, lemon, can, box, banana, strawberry, lego, cube, book, bowl, cup, mug, orange, lid, toy, wrench. - Receptacles: chair, black table, brown table, TV stand, sink, right counter, left counter, sofa, fridge, left drawer, right drawer, middle drawer. USER: <INSTRUCTION> ASSISTANT: ".

C.2 Action Space Adapter Details

We use the same ASA details between all environments. We detail the architecture and training decisions for the different ASAs when applicable.

VQ: Use a codebook size of 512 with 512 dimensions per codebook element. These 512 tokens are mapped to token indices $31000 - 31512$ from the LLaMA language modeling head. The encoder and decoder networks for predicting the latent and decoding from the latent are 4 layer MLP networks with hidden size 2048 using ReLU activations. The VQ network is trained on the actions in the same dataset used to train the policy. The network is trained with MSE loss to reconstruct the original actions. We VQ network for 3 epochs over the dataset.

RVQ: Use all the same details as VQ, but with a Residual-VQ that uses 2 codebooks.

Hyperparameter	CALVIN	Meta-World	BabyAI	HabPick
LR	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$	$3e^{-4}$
Optimizer	AdamW	AdamW	AdamW	AdamW
Number of Epochs	3	3	20	20
Batch Size Per GPU	32	32	8	32
Context Length	12	3	32	3
Max Gradient Norm	1	1	1	1

Table 3: Hyperparameters for all imitation learning experiments. Most hyperparameters are the same between environments but the number of training epochs, context length and batch size per GPU are adjusted to fit the need for history, environment dataset size and task complexity.

Pred: We use a 2 layer MLP network with a hidden size of 2048 and ReLU activations. We use this same MLP network architecture for discrete and continuous action space tasks. In the robot manipulation tasks, we also found it useful to include the robot proprioception as input to the MLP network and included this as input to the network layer. The robot proprioception consists of the robot robot joint angles and the gripper state. This ASA requires no separate training.

Uniform: In the tasks we consider, the actions are already normalized to be in $[-1, 1]$. We then create 512 evenly spaced bins within this interval and assign each action dimension based on which bin it is within. Like with VQ, we assign the 512 tokens to indices $31000 - 31512$ from the LLaMA language modeling head. This ASA requires no separate training.

Lang: Starting from the same semantic tokenization as with SemLang, we remap each token to the token corresponding to a digit "0" to "9". Therefore, the token count per action is the same between Lang and SemLang, but the Lang action tokens have no semantic meaning being just digits.

C.3 Training and Architecture Details

We use all pretrained components from LLaVA. For the visual token downsampler, we use a 2 layer Perceiver network [55] with 4 output latents and hidden size 4096.

We detail the hyperparameters used for imitation learning in in Table 3. We trained with the HuggingFace Transformers library [73], PyTorch [74], DeepSpeed [75]. For reinforcement learning, we use learning rate $3e^{-4}$, 32 steps per rollout. 18 parallel environment workers per GPU, an entropy coefficient of 0.01, 2 epochs over the data batch per rollout, 6 PPO minibatches, a maximum gradient norm of 0.2 and $\gamma = 0.99$.

We train the CALVIN, Meta-World and HabPick imitation learning results on a 4xA40 GPU setup. We train the Language Rearrangement and BabyAI experiments on a 8xA100-80GB GPU setup.

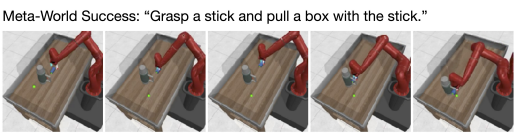
We train the LLM weights with LoRA and fine tune the entire ASA and downsampler module. For LoRA we use rank value 128, alpha parameter 32 and dropout 0.1.

D Qualitative Results

See Figure 8 for qualitative results of results from Figure 3. The RVQ ASA is visualized for Meta-World, CALVIN and Habitat Pick. SemLang is visualized for Language Rearrangement.

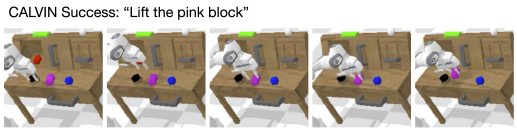
E Per-Task Breakdown

In this section, we show results for each environment by task type. Table 4 shows performance on Language Rearrangement for each of the evaluation datasets. Table 5 shows performance on CALVIN for each of the CALVIN tasks. Table 6 shows performance on BabyAI for each of the BabyAI instruction types. Table 7 shows performance on Meta-World for each of the 45 Meta-World task types.

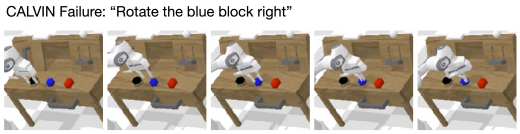


(a) The robot successfully picks the stick and pushes the box to the goal position.

(b) The robot only partially lifts the handle and fails to lift it up all the way.



(c) The robot grasps the pink block and lifts it to the goal height.



(d) The robot attempts to grasp the blue block but grasps too high, failing to pick the block.

(e) The robot moves closer to the cleaner bottle with its base and moves the arm to grasp the cleaner. It then returns the end-effector to the resting position to successfully end the task.

(f) The robot correctly finds the lemon in the sink, but the tight sink receptacle results in the arm colliding with the sink and the episode terminating due to excessive collisions.

(g) The robot searches the house, eventually finds the mug and then brings it to the blue table.

(h) The robot picks the strawberry navigates to the counter area, but puts the strawberry on the right counter as opposed to the correct receptacle of the sink.

Figure 8: Qualitative visualizations of successes and failures from the results in Figure 1 of the main paper. The RVQ action space adapter is visualized for Meta-World, CALVIN and Habitat Pick. SemLang is visualized for Language Rearrangement.

	Aggregated			Per Dataset Breakdown										
	Total	**Behavior Generalization**	**Paraphrastic Robustness**	**Train**	**Scene**	**Instruct Rephrasing**	**Novel Objects**	**Multiple Rearrange**	**Referring Expressions**	**Context**	**Irrelevant Text**	**Multiple Objects**	**Spatial**	**Conditional Instructs**
SemLang	51 $\pm$ 1	56 $\pm$ 2	47 $\pm$ 1	94 $\pm$ 3	94 $\pm$ 6	92 $\pm$ 1	97 $\pm$ 0	80 $\pm$ 6	31 $\pm$ 3	46 $\pm$ 14	66 $\pm$ 6	2 $\pm$ 2	0 $\pm$ 0	46 $\pm$ 4
Lang	27 $\pm$ 12	31 $\pm$ 14	24 $\pm$ 10	72 $\pm$ 13	58 $\pm$ 11	74 $\pm$ 12	76 $\pm$ 29	21 $\pm$ 10	10 $\pm$ 12	12 $\pm$ 11	20 $\pm$ 13	0 $\pm$ 0	2 $\pm$ 3	26 $\pm$ 16
Pred	42 $\pm$ 2	45 $\pm$ 3	38 $\pm$ 1	99 $\pm$ 1	96 $\pm$ 4	92 $\pm$ 2	95 $\pm$ 4	47 $\pm$ 5	26 $\pm$ 2	34 $\pm$ 2	32 $\pm$ 2	0 $\pm$ 1	8 $\pm$ 1	39 $\pm$ 3

Table 4: Evaluation results at 20M steps of RL training for all results in Language Rearrangement. We show averages and standard deviations over 2 random seeds of full policy training.

	RVQ	Pred	VQ	Uniform
CALVIN	72	68	56	28
turn off led	50	96	36	16
move slider left	99	100	100	15
rotate red block right	54	17	35	17
open drawer	100	100	56	100
rotate red block left	31	14	14	14
push pink block right	31	100	51	14
push blue block right	42	27	35	20
push red block left	68	36	61	17
push pink block left	47	50	86	14
push red block right	35	35	17	17
push blue block left	56	27	47	14
push into drawer	49	34	14	14
rotate pink block left	76	73	73	16
turn on lightbulb	80	34	19	9
rotate pink block right	30	73	19	10
rotate blue block right	28	13	13	13
turn off lightbulb	76	19	19	12
lift blue block table	34	25	34	16
close drawer	100	100	100	70
rotate blue block left	32	11	38	20
move slider right	100	100	100	19
turn on led	31	100	42	14
lift blue block slider	32	22	51	15
lift pink block table	66	68	82	11
lift red block slider	56	22	41	13
lift red block table	45	53	15	15
lift pink block slider	75	12	62	12

Table 5: Breakdown on every CALVIN task. Note there are not an equal proportion of all tasks in the evaluation dataset.

	SemLang	Lang	Pred
goto	90	90	75
pickup	60	35	35
open	26	7	21
putnext	8	5	7
pick up seq go to	21	12	22

Table 6: Breakdown on every BabyAI task.

	RVQ	Pred	VQ	Uniform
Meta-World	84	61	58	75
assembly	100	70	10	60
basketball	90	70	60	100
button-press-topdown	100	90	40	100
button-press-topdown-wall	100	100	60	90
button-press	100	100	70	100
button-press-wall	100	100	100	100
coffee-button	100	100	100	100
coffee-pull	100	40	30	50
coffee-push	80	20	30	80
dial-turn	100	50	40	100
disassemble	60	30	30	50
door-close	100	100	100	100
door-open	100	100	100	100
drawer-close	100	100	100	100
drawer-open	100	100	60	100
faucet-open	100	100	100	100
faucet-close	100	90	100	60
hammer	100	40	50	20
handle-press-side	100	100	100	100
handle-press	100	100	90	100
handle-pull-side	40	10	10	10
handle-pull	70	20	50	30
lever-pull	60	40	50	40
peg-insert-side	60	70	0	40
peg-unplug-side	50	30	100	90
pick-out-of-hole	50	90	40	30
pick-place	80	20	40	60
pick-place-wall	80	20	40	30
plate-slide	100	60	40	100
plate-slide-side	100	100	100	90
plate-slide-back	100	90	10	100
plate-slide-back-side	100	20	100	100
push-back	50	30	20	20
push	60	20	70	90
push-wall	80	40	60	100
reach	30	20	10	70
reach-wall	80	80	80	70
shelf-place	50	20	10	10
soccer	40	0	60	40
stick-push	100	60	10	100
stick-pull	90	50	30	100
sweep-into	70	40	60	70
sweep	90	30	50	70
window-open	100	100	100	100
window-close	100	100	100	100

Table 7: Breakdown on every Meta-World task.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, our claims in the abstract and introduce are experimentally verified in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, we mention limitations in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, we describe all experimental settings in detail in Appendix C.3, we build on open-source models and benchmarks and we plan to open-source the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We train and evaluate our method only on open benchmarks and train from an open source model. We also plan to release our code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Details provided in Appendix C.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: In our reinforcement learning results in Section 4.3 we show results over 2 seeds and display the standard deviation between seeds as a shaded region in the plot. For other experiments we only report results on 1 seed since we are finetuning 7B parameter models, making it computationally intensive to run multiple seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, complete details are proivded in Appendix C.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Yes, the paper adheres to the ethics code.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper is on foundational research not tied to any particular application and we do not feel it is important to highlight any negative societal impacts of our work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite all datasets used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: the paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.