

# DIFFUSION AUGMENTED AGENTS: A FRAMEWORK FOR EFFICIENT EXPLORATION AND TRANSFER LEARNING

Norman Di Palo

Imperial College London

London, UK

normandipalo@gmail.com

Leonard Hasenklever \*

Google DeepMind

London, UK

Jan Humplik \*

Google DeepMind

London, UK

Arunkumar Byravan \*

Google DeepMind

London, UK

## ABSTRACT

We introduce Diffusion Augmented Agents (DAAG), a novel framework that leverages large language models, vision language models, and diffusion models to improve sample efficiency and transfer learning in reinforcement learning for embodied agents. DAAG hindsight relabels the agent’s past experience by using diffusion models to transform videos in a temporally and geometrically consistent way to align with target instructions with a technique we call Hindsight Experience Augmentation. A large language model orchestrates this autonomous process without requiring human supervision, making it well-suited for lifelong learning scenarios. The framework reduces the amount of reward-labeled data needed to 1) finetune a vision language model that acts as a reward detector, and 2) train RL agents on new tasks. We demonstrate the sample efficiency gains of DAAG in simulated robotics environments involving manipulation and navigation. Our results show that DAAG improves learning of reward detectors, transferring past experience, and acquiring new tasks - key abilities for developing efficient lifelong learning agents. Supplementary material and visualizations are available on our website <https://sites.google.com/view/diffusion-augmented-agents/>.

## 1 INTRODUCTION

The most recent notable breakthroughs in AI have come from the combination of large models trained on enormous datasets (Firoozy et al., 2023; Brown et al., 2020; Hoffmann et al., 2022; Reed et al., 2022; Gemini-Team, 2023). However, despite efforts to scale up data collection (Collaboration, 2023; Reed et al., 2022; Bousmalis et al., 2023), data in embodied AI settings is still prohibitively scarce because such agents need to interact with physical environments where sensors and actuators present major bottlenecks (Cabi et al., 2020; Lee et al., 2022).

This data scarcity issue is especially pronounced in reinforcement learning scenarios, where rewards are often sparse or completely absent in realistic settings (Ecoffet et al., 2021). Overcoming these challenges requires developing agents that can learn and adapt efficiently from limited experience. We hypothesize that embodied agents can achieve greater data efficiency by leveraging past experience to explore effectively and transfer knowledge across tasks (e.g. (Andrychowicz et al., 2017)). In particular, we are interested in enabling agents to autonomously set and score subgoals, even in the absence of external rewards, and to repurpose their experience from previous tasks to accelerate learning of new tasks.

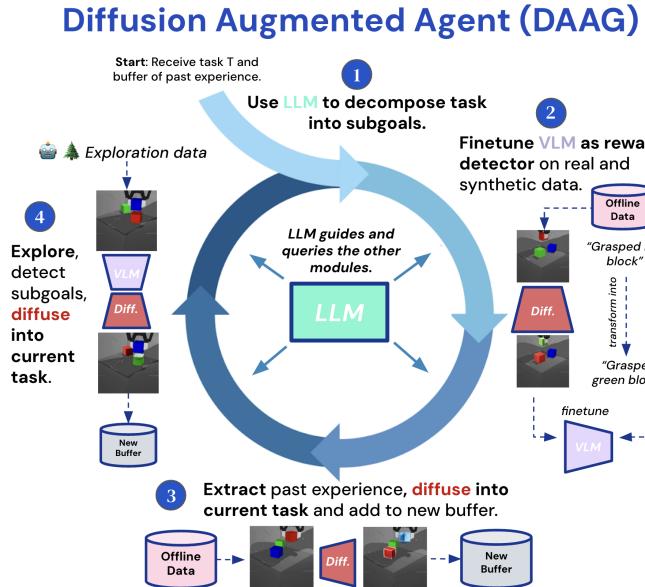


Figure 1: An illustration of our proposed framework.

\* Senior Author

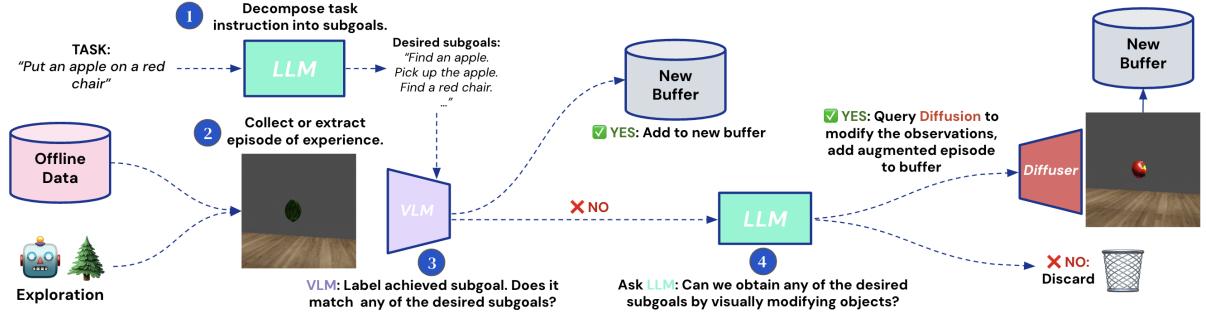


Figure 2: An example of the Hindsight Experience Augmentation pipeline, with which new or past experience can be optionally modified and added to a new buffer.

In this paper, we address these questions using foundation models pretrained on internet-scale datasets (Firoozi et al., 2023). Through an interplay of vision, language, and diffusion models (Radford et al., 2021; Alayrac et al., 2022; Brown et al., 2020; Rombach et al., 2022), we enable the agent to more effectively reason about its tasks, interpret its environment and past experience, and manipulate the data it collects to repurpose it for novel tasks and objectives. Importantly, DAAG operates autonomously without the need for human supervision, making it particularly well-suited for lifelong reinforcement learning scenarios.

In Figure 1 we illustrate our framework from a high-level perspective. A large language model (LLM) acts as the main controller, or *brain*, querying and guiding a vision language model (VLM) and a diffusion model (DM) and the high-level behavior of the agent. We call our framework **Diffusion Augmented Agent (DAAG)**.

Through a series of experiments on different environments, including robot manipulation and navigation, we empirically demonstrate that our proposed framework improves the agent’s performance on a series of key abilities: **1) autonomously computing rewards** for both seen and unseen tasks by fine-tuning a vision language model on data augmented with synthetic samples generated by a diffusion model; **2) more efficiently exploring and learning new tasks** by designing and recognising useful sub-goals for the given task, and repurposing otherwise failed trajectories by modifying the recorded observations via diffusion models; **3) effectively transferring previously collected data to new tasks** by both extracting related data and repurposing other trajectories through the use of diffusion models. In Figure 2, we illustrate how our method can repurpose agent’s experience via diffusion augmentation. We propose a diffusion pipeline that improves geometrical and temporal consistency to modify part of videos collected by agents (Fig. 4), a novelty in the field of reinforcement learning to the best of our knowledge.

## 2 RELATED WORK

**Foundation Models in Embodied AI** The rapid evolution of foundation models (Bommasani et al., 2022) such as large language models (LLMs) (Brown et al., 2020; Hoffmann et al., 2022; Gemini-Team, 2023) and vision language models (VLMs) (OpenAI, 2023; Gemini-Team, 2023; Alayrac et al., 2022; Radford et al., 2021) has sparked significant interest from the robotics and embodied AI research community in recent years. As these models have demonstrated increasingly sophisticated abilities in language understanding, reasoning, commonsense knowledge, and visual perception, researchers have leveraged their potential as building blocks for intelligent agents.

Many methodologies for integrating these models into robotics systems were proposed in recent months. (Wang et al., 2023; 2024; Firoozi et al., 2023). (Ahn et al., 2022; Liang et al., 2023; Di Palo et al., 2023; Huang et al., 2022) proposed the use of LLMs as high-level planner, able to decompose an instruction into a series of short horizon sub-goals. These models than employ different modalities to ground such textual plans into actions. (Huang et al., 2023; Yu et al., 2023b) use LLMs and VLMs to instead obtain a reward function, given a textual instruction, that can be then used by an external optimiser to compute a trajectory. (Brohan et al., 2023; Kwon et al., 2023) use vision and language models to directly output executable actions given a textual description of a task. (Xiao et al., 2023; Di Palo et al., 2023) use VLMs as reward detectors/task classifiers. In this work, we build upon the framework proposed in (Di Palo et al., 2023), using an LLM to decompose long horizon plans into a sequence of subgoals. However, we extend it by giving the LLM the ability to query a diffusion model to modify and augment visual observations autonomously, unlocking faster learning and more effective transfer.

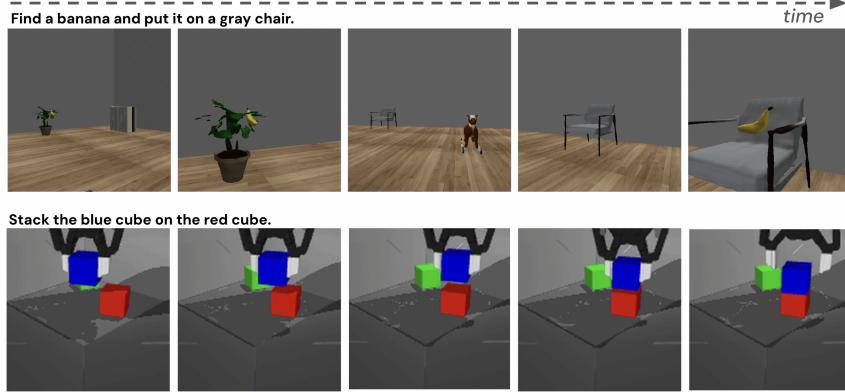


Figure 3: Two example instances of tasks we use in our investigation.

**Image Generation with Diffusion Models** The rapid evolution of generative AI, beyond language-based application, also saw the exponential growth of image generation models, largely based on diffusion processes (Dhariwal and Nichol, 2021; Rombach et al., 2022; Ho et al., 2020; Ramesh et al., 2021). Mostly, these models are conditioned via text, specifying the desired output image in natural language. Beyond generating images from scratch, these models have also been employed to modify images via in-painting, i.e. the completion of a specific, masked part of an original image. To provide more fine-grained control of the generation process, recent works have enabled diffusion models with the ability to be conditioned on modalities other than text: (Zhang et al., 2023) demonstrated the use of depth maps, canny edges, segmentation masks and more in addition to textual instructions to guide and constrain the geometrical and visual aspect of the final outputs. We largely based our proposed diffusion pipeline on these results, as ensuring the augmented observations respect the original geometries of objects and environment is fundamental in embodied applications. Image-based diffusion models have also been recently extended into the temporal domain, outputting short videos (Blattmann et al., 2023) instead of single frames. Training such models is however particularly challenging, and they do not yet benefit from the fine grained control abilities mentioned above. Notably, (Khachatryan et al., 2023) demonstrated that image diffusion models can be repurposed to generate videos simply by constraining the original noise maps used to start the backward diffusion process.

**The Use of Diffusion Models in Robotics** While notable research has been conducted on the use of diffusion models as a way to represent policies (Chi et al., 2023) or generate additional experience for the agent as low-level states (Lu et al., 2023), in this work we will focus on their use in the visual domain as text-conditioned image generators. As visual perception is fundamental for robotics and embodied AI, the recent literature investigated methodologies to integrate the image generation abilities of such models to empower agents (Zhu et al., 2024). To augment the visual experience collected by a robot, (Chen et al., 2023; Mandi et al., 2023; Yu et al., 2023a) propose the use of diffusion models to either modify the background to increase robustness to distractors, or repurpose trajectory for new tasks by modifying the manipulated objects into new ones of interest (Yu et al., 2023a). While the latter is similar to our proposed approach, there are some fundamental differences: (Yu et al., 2023a) is based on an imitation learning pipeline, where a human operator labels the task demonstrated and the new task to generate via diffusion. Our method, designed and revolving around the abilities of LLMs, is entirely autonomous both in terms of detecting the accomplished tasks (via the VLM) and proposing and generating augmentations (via the LLM querying the diffusion model), therefore being more suited for (lifelong) reinforcement learning scenarios, where human supervision is not always present. Additionally, we propose a diffusion pipeline that is both geometrically and temporally consistent when modifying video, differently from (Chen et al., 2023; Mandi et al., 2023; Yu et al., 2023a). Ensuring temporal and geometrical consistency is beneficial when dealing with embodied environments, as we later demonstrate.

**Experience Re-Use and Transfer** The need for vast amounts of experience data to learn policies (Reed et al., 2022; Bousmalis et al., 2023) inspired the research community to propose methods and strategies to re-use experience collected for different, previous tasks. Hindsight Experience Replay (Andrychowicz et al., 2017) proposes a method to re-use episodes that solve tasks different from the desired task. If asked to solve task  $\mathcal{T}_n$ , but solving instead task  $\mathcal{T}_m$  during exploration, (Andrychowicz et al., 2017) proposed to relabel the episode as if the desired task was  $\mathcal{T}_m$ , therefore re-purposing a trajectory as a success for a different task. Differently, when in the same situation, our method modifies the collected observations that solve task  $\mathcal{T}_m$  to synthetically generate an episode that would have solved the desired task  $\mathcal{T}_n$ . This means we can generate data for a task *without the need to effectively solve it during exploration*. (Bousmalis et al., 2023) repurposes data from various task by learning a single, goal-conditioned policy, therefore

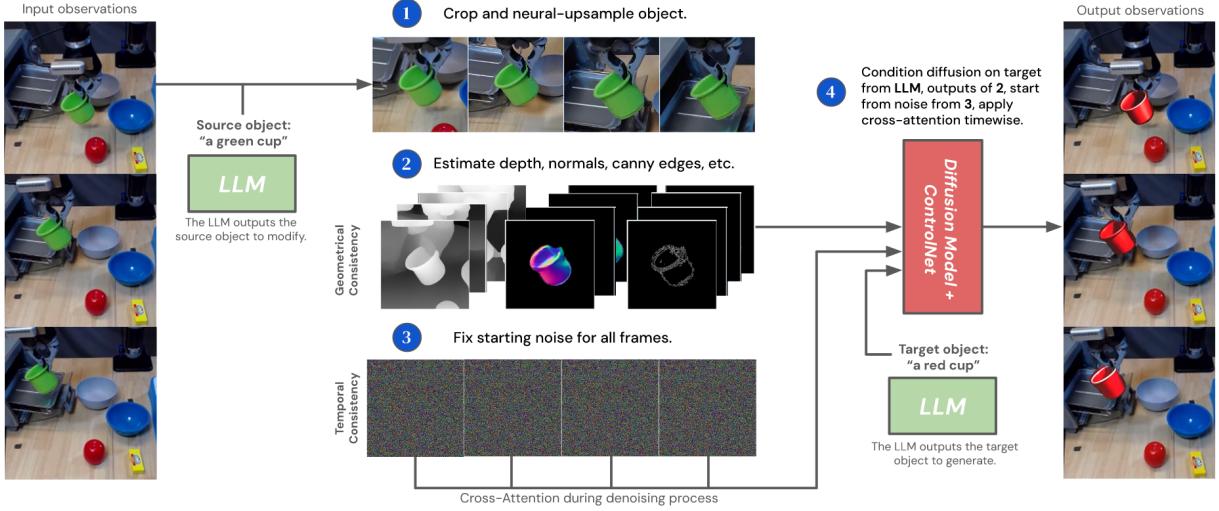


Figure 4: An illustration of our diffusion pipeline, highlighting the geometrical and temporal consistency obtained by combining the methodologies in (Zhang et al., 2023) and (Khachatryan et al., 2023).

extracting learning signal from any episode solving any task. However, in our experiments we demonstrate that trying to learn a single, goal-conditioned policy is not as effective as explicitly generating and training on data for the specific task we are interested in. (Tirumala et al., 2023) demonstrated the effectiveness of re-using collected experience from past runs and experiments to bootstrap learning: in this work, we first extract only relevant data, as in (Di Palo et al., 2023), through the use of vision and language models, and additionally generate bespoke new data from past data for the new task at hand, demonstrating improved learning efficiency.

### 3 METHOD

#### 3.1 PRELIMINARIES

We formalise our environments as Markov Decision Processes (MDPs): the environment and the agent, at each timestep  $t$ , are in a state  $s \in \mathcal{S}$ . From that state, the agent receives a visual observation  $o \in \mathcal{O}$ , and can execute an action  $a \in \mathcal{A}$ . During each episode, the agent receives an instruction, which is a description of the task to execute in natural language  $\mathcal{T}$ . The agent can receive a reward  $r = +1$  at the end of the episode if the task is successfully executed.

In this work, beyond learning new tasks in isolation, we study our framework’s ability to learn tasks in succession in a lifelong fashion. Therefore, the agent stores interaction experiences in two buffers: the current task buffer that we call *new buffer*  $\mathcal{B}_n$ : this buffer is initialised at the beginning of each new task. There is then an *offline lifelong buffer*  $\mathcal{B}_{ll}$ : the agent stores all episodes from all tasks in this buffer, regardless of their success. The latter is therefore an ever-growing buffer of experiences the agent can then use to bootstrap learning of new tasks.

**Large Language Model:** We use a large language model to orchestrate the behaviour of the agent and the use of the vision language model (VLM) and diffusion model. The LLM receives a textual instruction and data and outputs textual responses. In our work, we leverage the LLM’s ability to decompose tasks into subgoals, compare the similarity of different tasks/instructions, and query the VLM and diffusion model. We parse the output of the LLM to obtain the exact string we need for each use case. In the Supplementary Material on our website, we show the way we design the prompt to guide the textual generation of the LLM and simplify the final parsing of its output.

**Vision Language Model:** The VLM we use is **CLIP** (Radford et al., 2021), a contrastive model. CLIP is composed of two branches: an image branch  $\phi_{\text{image}}$  and a textual branch  $\phi_{\text{text}}$ . They respectively take as input visual observations and textual descriptions, outputting embedding vectors of the same size  $y_{t,\text{im}} = \phi_{\text{image}}(o_t)$ ,  $y_{g,\text{txt}} = \phi_{\text{text}}(\mathcal{T}_g)$ . The peculiarity of the output embeddings is the following: their cosine similarity  $s_{g,t} = cs(y_{t,\text{im}}, y_{g,\text{txt}})$  implicitly represents how well the text  $\mathcal{T}_t$  describes the observation  $o_t$ . Following (Di Palo et al., 2023), we consider that the VLM labels an observation  $o_t$  as being a goal observation for task  $\mathcal{T}_g$  if  $s_{g,t} > \delta$ , where  $\delta$  is a threshold computed during training. More details are presented in the Supplementary Material on our website. Therefore, given a set of goal tasks  $\mathcal{T}_{0:G}$ , for each

new observation CLIP computes a new  $s_{g,t} = cs(y_{t,im}, y_{g,txt})$  and if the threshold is surpassed, labels the observation as achieving the task at hand.

As CLIP needs to be explicitly given as an input a textual description, in addition to the image, we need to manually provide a set of tasks we are interested in detecting while the agent is exploring. Briefly, we adopt two strategies: first, we keep a list of all the tasks that were given to the agent up to that point, and all the relative subgoals obtained by the LLM. Additionally, we can autonomously propose possible subgoals by giving to the LLM the aforementioned list and a list of objects present in the environment. These techniques allow us to have a list of tasks that the agent may randomly achieve at test time.

**Diffusion Pipeline:** A core aspect of our work is modifying visual observations through language-instructed diffusion models. The goal of our diffusion pipeline is to take an observation  $o_t$  or a temporal series of observations recorded by the agent  $o_{t:t+H}$  and visually modify one or more objects present in the observation(s), while enforcing geometrical and temporal consistency.

To tackle the former, the diffusion pipeline receives as input the observation(s) to be modified, a textual description of the object to modify, or *source object*  $obj_s$ , and a description of the object to generate in place of the source, or *target object*  $obj_t$ . The former is used to localise, crop and mask the source object to perform inpainting, while the latter is used by the diffusion model itself to guide the image generation. The pipeline receives, additionally, a series of visual inputs to condition the image generation, all computed from the original observation(s): depth maps  $o_{\text{depth}}$ , canny edges  $o_{\text{canny}}$ , and normals maps  $o_{\text{normals}}$ . We compute these from RGB observations via a series of off-the-shelf models we list in the Supplementary Material. The latter are used, through the ControlNet architectures (Zhang et al., 2023), to ensure that the generated objects respect the geometry of the original objects. While the use of each is optional, their combined use improves the overall results. The overall process can be described as  $\hat{o}_{t:t+H} = \text{Diff}(o_{t:t+H}, obj_s, obj_t, o_{\text{depth}}, o_{\text{canny}}, o_{\text{normals}})$ , where  $\hat{o}_t$  represents an observation modified via diffusion from the original observation  $o_t$ . Fig. 4 represent an illustration of the entire pipeline.

To improve temporal consistency, we apply the technique proposed in (Khachatryan et al., 2023): when applying diffusion to  $N$  frames, we 1) fix the initial noise map for all  $N$  instead of sampling different ones (the results will still be different as the ControlNet inputs will be different), and 2) add **temporal cross-attention** to the diffusion process: all the frames can therefore attend to all other frames during the backward diffusion process for image generation. This does not require any architectural change, nor retraining the model. In Figure 5, we provide a visual comparison of the outputs of the method proposed in (Yu et al., 2023a) and our proposed pipeline. The former does not keep object poses and aspect consistent over frames, therefore invalidating the hypothesis that applying the same actions  $a_{0:T}$  to the modified observations  $\hat{o}_{0:T}$  would have led to successful completion of the new task.

Now that we introduced the main components of DAAG, we will describe the way they interoperate in our framework.

### 3.2 FINETUNE, EXTRACT, EXPLORE: THE DIFFUSION AUGMENTED AGENT FRAMEWORK

**Finetuning VLMs as Reward Detectors on Diffusion Augmented Data:** VLMs can be effectively employed as reward detectors, conditioned on a language-defined goal and a visual observation. However, as demonstrated by recent works (Di Palo et al., 2023; Xiao et al., 2023), to be accurate they often need to be finetuned on labelled data gathered in the target environment, for the desired tasks. This is a time-consuming task that furthermore requires human effort for each new task to be learned, hindering the ability of the agent to autonomously learn many tasks in succession in a lifelong fashion. With our framework we tackle this challenge by finetuning the VLM on previously collected observations. Given a dataset  $\mathcal{D}$  of observations  $o_i$ , each paired with a label  $\mathcal{T}_i$ , and a new goal task expressed in natural language,  $\mathcal{T}_g$ , we extract all observations whose caption  $\mathcal{T}_i$  is similar enough to  $\mathcal{T}_g$ , or such that a visual modification of the corresponding observation  $o_i$  would transform it into a fitting observation  $\hat{o}_i$  for  $\mathcal{T}_g$ : for example, given the goal description “*The robot is grasping the red cube*”, an observation with caption “*The robot is grasping the blue cube*” can be modified by visually swapping *red cube* with *blue cube* through a controlled diffusion process. In DAAG, the LLM

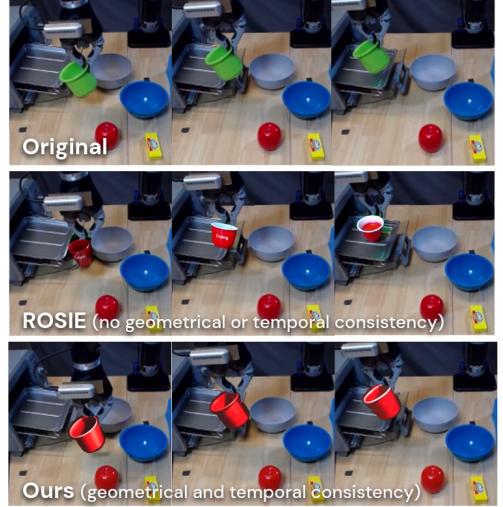


Figure 5: Outputs of the method proposed in ROSIE (Yu et al., 2023a) and our diffusion pipeline when asked to “swap a green cup with a red cup”.

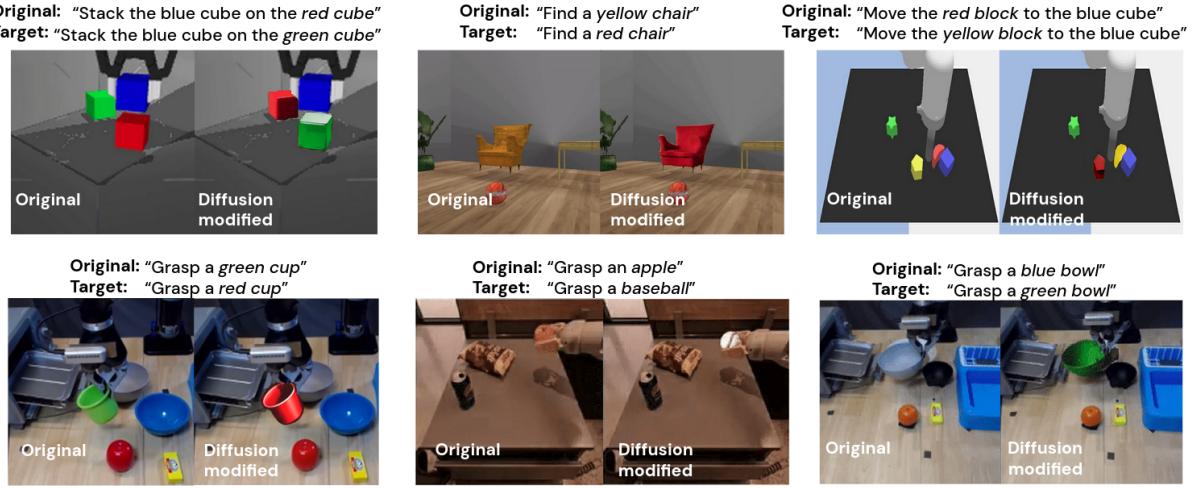


Figure 6: Examples of original observations and diffusion modified observations, showing the original achieved goal and the desired goal, taken from our environments and from the Open X-Embodiment dataset (Collaboration, 2023).

autonomously selects the fitting observations  $o_i$  from  $D$  by comparing their caption  $\mathcal{T}_i$  with the goal caption  $\mathcal{T}_g$ : if a swap is considered possible, the LLM instructs the DM with the source object to modify and the target object to add (in the previous example  $\{\text{blue cube, red cube}\}$  respectively). This process is illustrated in Figure 2. Through this process, we finetune the VLM to act as a success detector for all the subgoals  $\mathcal{T}_{0:G}$  in which the task at hand was decomposed by the LLM.

**Efficient Learning and Transfer via Hindsight Experience Augmentation:** After each episode collected on any task it encounters, an agent collects a series of observations and actions  $E_n = \{o_t, a_t\}_{t=0}^T$ . We store every episode, regardless of the final outcome, in the lifelong buffer  $\mathcal{B}_{ll}$  (Cabi et al., 2020). When learning a new task, the agent receives a task instruction in textual form  $\mathcal{T}_g$  and decomposes it into a series of subgoals  $\mathcal{T}_{0:G}$  via the LLM. Normally, the agent can extract a learning signal only from episodes that are collected via exploration or from past experience stored in  $\mathcal{B}_{ll}$  if there are rewards associated to it; these rewards can be either from the VLM or be an external reward from the environment. In DAAG, we aim to maximise the number of episodes from which the agent can learn to tackle a new task, even if it does not achieve any of the desired subgoals. We do this through a process we call **Hindsight Experience Augmentation (HEA)**.

Given an episode of experience  $E_i = \{o_t, a_t\}_{t=0}^T$ , we use the VLM to label what possible subgoals have been achieved by the agent, as described in the Preliminaries (more details on this phase are presented in the Suppl[‘900] -imentary Material). If any matches a desired subgoal, we add this episode to the new, current task buffer  $\mathcal{B}_n$ . This process emulates the framework proposed in (Di Palo et al., 2023). However, if no match is present, instead of discarding the episode, we query the LLM to ask if the achieved subgoal(s) can match any of the desired subgoals by swapping/visually modifying any of the objects, e.g. matching “*The red cube is stacked on the blue cube*” with “*The green cube is stacked on the blue cube*” by swapping *red cube* with *green cube*. When a swap is identified as possible, the LLM queries the diffusion model to modify the observations of the episode up to the achieved sub-goal  $[o_0, \dots, o_{T_g}]$  into  $[\hat{o}_0, \dots, \hat{o}_{T_g}]$ , and finally adds the modified observations and the original actions in the experience buffer  $\mathcal{B}_n \leftarrow [(\hat{o}_0, a_0), \dots, (\hat{o}_{T_g}, a_{T_g})]$ .

Through HEA, we can synthetically increase the number of successful episodes the agent can store in its buffers and learn from. This allows to effectively re-use as much data gathered by the agent as possible, substantially improving efficiency especially when learning multiple tasks in succession, as we will describe later. While previous methods have proposed the use of diffusion-based image generation or augmentation to synthesise additional data for learning policies (Yu et al., 2023a; Mandi et al., 2023; Chen et al., 2023), our method is the first to propose an entire autonomous pipeline, independent from human supervision, and that leverages geometrical and temporal consistency to generate consistent augmented observations.

When learning a new task, the agent first applies HEA to all the episodes stored in its lifelong buffer  $\mathcal{B}_{ll}$  to start with a non-empty new task buffer  $\mathcal{B}_n$ . It then trains a policy on this data to kickstart exploration, and subsequently applies HEA to any new episode of experience gathered via exploration. By dividing a goal into shorter-horizon subgoals via the LLM, and more efficiently learning to tackle those via HEA, our agent quickly learns to guide exploration, as it learns to solve the various subgoals that lead to the completion of the task. In a robotic stacking scenario, for example,

DAAG efficiently learns to pick up the first object to stack. By consistently solving that first step during exploration, it is more likely to also randomly complete the task by placing it on top of the target object.

We will shed light on the effect of both of these phases on learning efficiency in the Experiments section. The entire pipeline is illustrated in Fig. 2.

## 4 EXPERIMENTS

Our framework, **DAAG**, proposes an interplay between LLMs, VLMs and diffusion models to tackle three principal challenges in agents that learn in a lifelong fashion: **1)** finetuning a new reward/sub-goals detection model, **2)** extracting and transferring past experience for new tasks and **3)** efficiently exploring new tasks. To thoroughly investigate the benefits of DAAG on these three challenges, we designed and ran a series of experiments that individually measure the contribution and benefits of our method on each of these scenarios. The rest of the section is therefore divided into three main subsections, where each of these challenges is investigated and results are demonstrated and analysed: **1)** can DAAG finetune VLMs as reward detectors for novel tasks? **2)** can DAAG explore and learn new tasks more efficiently? **3)** can DAAG more effectively learn tasks in succession, transferring experience from past tasks?

### 4.1 EXPERIMENTAL SETUP

We use three different environments to measure the performance of DAAG on the aforementioned challenges: **1)** a robot manipulation environment, **RGB Stacking** (Lee et al., 2022), where a robot arm is tasked with stacking colored cubes into a goal configuration. The action space  $\mathcal{A}$  is composed of a target and goal pick and place position, represented as a pair of  $\mathbb{R}^2$  numbers, and the observation space  $\mathcal{O}$  is composed of RGB visual observations captured from a fixed shoulder camera. **2)** a navigation environment, **Room**, inspired by (Team et al., 2022), where an agent navigates in a room filled with objects and furniture, and is tasked with picking up and placing a goal object on a goal chair. The action space  $\mathcal{A}$  is composed of a forward velocity and rotational velocity input represented as  $\mathbb{R}^2$ . We assume the agent can pick up an object automatically by moving in close proximity to it, and equally place it on a target object when sufficiently close to it. The observation space  $\mathcal{O}$  is composed of RGB visual observations captured from a first-person view camera. **3)** a non-prehensile manipulation environment, **Language Table** (Lynch et al., 2022), where a robot can push colored blocks on a table to move them to a goal configuration. The action space  $\mathcal{A}$  is composed of  $x$  and  $y$  end-effector velocity inputs as  $\mathbb{R}^2$ . The observation space  $\mathcal{O}$  is composed of RGB visual observations captured from a fixed shoulder camera. Goals for all environments are provided as *natural language instructions*.

As a policy learning algorithm, we use Self-Imitation Behavior Cloning (Chen et al., 2021; Di Palo et al., 2023; Oh et al., 2018) on all the episodes stored in the buffer  $\mathcal{B}_n$ , where the agent collects all successful episodes.

As a large language model, we use Gemini Pro (Gemini-Team, 2023). As a VLM, we use CLIP ViT-B/32 (Radford et al., 2021). As a diffusion model, we use Stable Diffusion 1.5 (Rombach et al., 2022), with ControlNet (Zhang et al., 2023).

Detailed hyperparameters and values of constant are listed in the Supplementary Material.

### 4.2 CAN DAAG FINETUNE VLMs AS REWARD DETECTORS FOR NOVEL TASKS?

In this section we evaluate the ability of DAAG to obtain effective reward detectors for new tasks by finetuning VLM on past experiences collected by the agent being augmented through a diffusion pipeline, as explained in 3.

We assume the existence of a dataset  $\mathcal{B}$  of collected goal observations for different tasks  $\mathcal{T}_{\mathcal{B}} = [\mathcal{T}_0, \dots, \mathcal{T}_n]$ . We then want to measure the performance of CLIP, the VLM we use in this work, to correctly detect novel observations  $o_t$  as goal configurations for a new task  $\mathcal{T}_i$  not present in  $\mathcal{T}_{\mathcal{B}}$ . We compare finetuning CLIP on the original dataset  $\mathcal{B}$ , and finetuning on an artificially expanded version of  $\mathcal{B}$  where we apply diffusion augmentation to synthesise examples of goal observations for  $\mathcal{T}_i$ , starting from goal observations of other tasks.

To empirically evaluate if the synthetic observations positively affect performance, we compare the two pipelines on a test set of unseen observations, where 50% are goal observations of  $\mathcal{T}_i$  and 50% are not, therefore resulting in a balanced binary classification task. We also compare the zero-shot performance of CLIP, to better evaluate the relative improvement over the off-the-shelf model.

For the **RGB Stacking** environment, the tasks  $\mathcal{T}_{\mathcal{B}}$  are [ "Stack the green cube on the blue cube", "Stack the blue cube on the red cube" ], respectively  $\mathcal{T}_{g,b}^{\text{RGB}}$ ,  $\mathcal{T}_{b,r}^{\text{RGB}}$ , with the test task  $\mathcal{T}_i$  being "Stack the red cube on the green cube",  $\mathcal{T}_{r,b}^{\text{RGB}}$ .

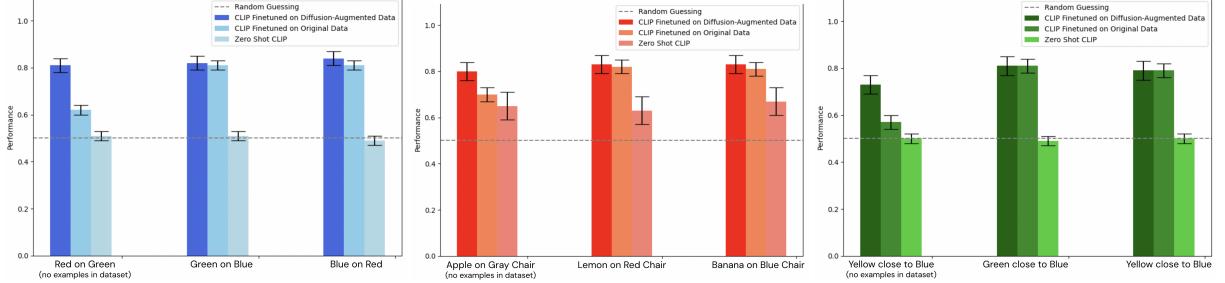


Figure 7: Performance of a finetuned CLIP as a reward detector, evaluating the use of synthetic observations to detect reward of a new, unseen task. In all three bar plots, the leftmost task is an held-out task, for which there are no examples in the dataset we use to finetune CLIP. DAAG can synthetically generate data to successfully finetune CLIP as a reward detector, while not affecting the performance on the other tasks. We plot mean and standard deviation by repeating the experiments with three different training and test sets per environment.

For the **Room** environment, the tasks are [ “Put a lemon on a red chair”, “Put a banana on a blue chair” ], respectively  $\mathcal{T}_{\text{Room}}^{\text{lemon,red}}, \mathcal{T}_{\text{Room}}^{\text{banana,blue}}$ , with the test task  $\mathcal{T}_i$  being “Put an apple on a gray chair”,  $\mathcal{T}_{\text{apple,gray}}$ .

For the **Language Table** environment, the tasks are [ “Put the green block near the blue block”, “Put the yellow block near the blue block” ], respectively  $\mathcal{T}_{g,b}^{\text{LT}}, \mathcal{T}_{y,b}^{\text{LT}}$ , with the test task  $\mathcal{T}_i$  being “Put the red block near the blue block”,  $\mathcal{T}_{y,b}^{\text{RGB}}$ .

For each environment, we have  $N = 100$  example observations for each  $\mathcal{T}_B$ , and use the DAAG pipeline to obtain synthetic observations of  $\mathcal{T}_i$  by augmenting all other observations. We finetune CLIP on both the original and augmented datasets and test for accuracy on a test set of  $M = 100$  unseen observations. We report results in Figure 7. The results demonstrate how, in each environment, DAAG can learn an effective reward detector even when having no example observations of such task, *outperforming a CLIP model trained on the other tasks and queried to generalise zero-shot to the new task*. Figure 7 shows how, on the leftmost task that has no examples in the dataset, DAAG brings a substantial improvement by synthesising examples from other tasks, while keeping the same performance on the seen tasks. In the RGB Stacking and Language Table environments, where precise geometric relations between objects poses are fundamental, the difference with the baselines is more impressive, shedding light on the need for diffusion augmentation to obtain an effective reward detector. In the Room environment, the observations CLIP receives, albeit coming from a low-fidelity simulator and renderer, are closer to the distribution of observations it received during training on a web-scale dataset (pictures of fruits and furniture). Therefore, zero-shot performance is considerably stronger there, while in the other tasks is close to random guessing, demonstrating the need for finetuning.

#### 4.3 CAN DAAG EXPLORE AND LEARN NEW TASKS MORE EFFICIENTLY?

We here focus on investigating the benefits brought by DAAG and HEA to exploring and learning new tasks from scratch.

We assume the agent start learning a new task *tabula rasa* in this experimental scenario, with an empty new task buffer  $\mathcal{B}_n$ , and with no access to a lifelong buffer  $\mathcal{B}_{ll}$ , to independently evaluate the effect of HEA on new task learning efficiency. The agent receives a natural language instruction  $\mathcal{T}_i$ , that informs it about the goal to achieve in the environment. We experimentally evaluate the learning efficiency improvements brought by HEA, comparing against (Di Palo et al., 2023), that decomposes  $\mathcal{T}_i$  into subgoals  $\mathcal{T}_{0:G}$  and obtained reward for each via a VLM, and to a baseline agent that does not benefit neither from task decomposition or diffusion augmentation.

We evaluate the performance of this method on the RGB Stacking and Room environments. For the **RGB Stacking** environment, the task is “Stack the red cube on the blue cube”, or  $\mathcal{T}_{r,b}^{\text{RGB}}$ . For the **Room** environment, the task is *Put an apple on a gray chair*,  $\mathcal{T}_{a,g}^{\text{Room}}$ . The environments only provide a reward of +1 when the task is successfully achieved, and end the episode there: in that case, we add the entire episode to  $\mathcal{B}_n$ . If an internal reward is detected via the VLM at timestep  $T_g$ , the observations and actions up to that timestep are added to  $\mathcal{B}_n$ . We perform exploration in

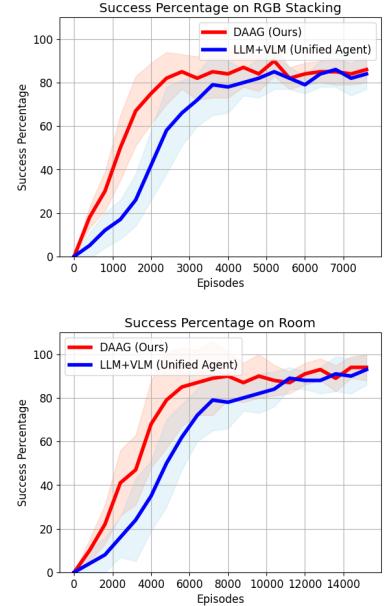


Figure 8: Performance of learning new tasks from scratch on RGB Stacking and Room. In the plot we show mean and standard deviation over 3 seeds.

the environments via an epsilon-greedy strategy (Mnih et al., 2013). We start with  $\epsilon = 0.99$  and decay it over time, with detailed hyperparameters described in the Supplementary Material. During each episode, if we sample a number  $n > \epsilon$  where  $n \sim \mathcal{U}(0, 1)$ , we let the policy network guide the agent. Otherwise, we perform exploration as follows: for the RGB Stacking task, we sample a random pick up position  $a_0 \in \mathbb{R}^2$  and a random place position  $a_1 \in \mathbb{R}^2$  and execute the actions. For the Room task, in order to speed up and guide exploration, we select a random pickable object in the room, and move the agent to it to pick it up, collecting all observations and actions leading to it  $[o_t, a_t]_{t=0}^{T_{\text{pick}}}$ , with  $a_t \in \mathbb{R}^2$ . We then select a random target furniture where to place the object, and move the agent there collecting other observations and actions  $[o_t, a_t]_{t=T_{\text{pick}}}^{T_{\text{place}}}$ . While speeding up the learning of the task, this guidance does not affect the relative performance improvements brought by one method over the other: in the Supplementary Material, we also show training curves with entirely random exploration. We train our policy every  $T_{\text{BC}}$  steps on the observation-action pairs collected in  $\mathcal{B}_n$ .

In Figure 8, we plot the number of successfully solved instances of the task over 100 test episodes as a function of the number of training episodes. During testing, we do not perform any exploration strategy or guidance, and let the policy network guide the agent. We can see how DAAG learns faster than the baseline. The ability to use even certain unsuccessful episodes as learning signal helps improving the learning efficiency across all tested environments.

#### 4.4 CAN DAAG MORE EFFECTIVELY LEARN TASKS IN SUCCESSION TRANSFERRING EXPERIENCE FROM PAST TASKS?

We now investigate the influence of DAAG on another fundamental ability of lifelong learning agents: the ability to extract, transfer and repurpose past experience to speed up learning of new tasks. As we demonstrated the ability of DAAG to learn new tasks efficiently through exploration and HEA in the previous set of experiments, we now investigate its ability to extract information and learn policies from a given dataset of experience, with no additional exploration, in a setting closer to Offline Reinforcement Learning (Kostrikov et al., 2021).

We let an agent learn three tasks in sequence per environment. At the beginning of each new task  $\mathcal{T}_n$ , the agent receives a buffer of experience  $\mathcal{B}_{ll,n}$  containing  $N_{\text{off}} = 200$  episodes composed as follows: 50% are successful episodes of the task at hand  $\mathcal{T}_n$ , while the other 50% are episodes solving different tasks in the same environment. Each baseline uses all buffers and data received up to that point  $\mathcal{B}_{ll,0:n}$  to learn a policy and is then tested on 100 test episodes to solve the task  $\mathcal{T}_n$ : for task  $n$  the agent has therefore access to  $n \times N_{\text{off}}$  pre-collected episodes.

When learning to solve task  $\mathcal{T}_n$  using  $\mathcal{B}_{ll,0:n}$ , (Di Palo et al., 2023) decomposes  $\mathcal{T}_n$  into subgoals  $\mathcal{T}_{0:G}$  and for each extracts successful trajectories from  $\mathcal{B}_{ll,0:n}$ . DAAG, in addition to this, runs *Hindsight Experience Augmentation* to also extract trajectories completing similar tasks that can be visually modified to match any of the subgoals in  $\mathcal{T}_{0:G}$ . Both baseline train the Self-Imitation Learning policy on the extracted (and synthetically augmented) episodes. In addition, we run another baseline which does not perform any decomposition or extraction, and only trains a goal-conditioned policy on all the episodes contained in  $\mathcal{B}_{ll,0:n}$ .

For each environment, we learn three tasks in succession. For the RGB Stacking environment, the tasks are, in order, [ “Stack the red cube on the green cube”, “Stack the green cube on the blue cube”, “Stack the blue cube on the red cube” ], respectively  $\mathcal{T}_{r,g}^{\text{RGB}}, \mathcal{T}_{g,b}^{\text{RGB}}, \mathcal{T}_{b,r}^{\text{RGB}}$ .

For the **Room** environment, the tasks are [ “Put a lemon on a red chair”, “Put a banana on a blue chair”, “Put an apple on a gray chair” ], respectively  $\mathcal{T}_{\text{lemon},\text{red}}^{\text{Room}}, \mathcal{T}_{\text{banana},\text{blue}}^{\text{Room}}, \mathcal{T}_{\text{apple},\text{gray}}^{\text{Room}}$ .

In Figure 9, we compare the performance, as success rate, of each method on method  $\mathcal{T}_n$  using  $\mathcal{B}_{ll,0:n}$ . We can see how DAAG surpasses both baselines, thanks to the ability to learn from most of the experience stored in  $\mathcal{B}_l$ , by modifying and repurposing trajectories solving tasks beyond  $\mathcal{T}_n$  or its subgoals  $\mathcal{T}_{0:G}$ .

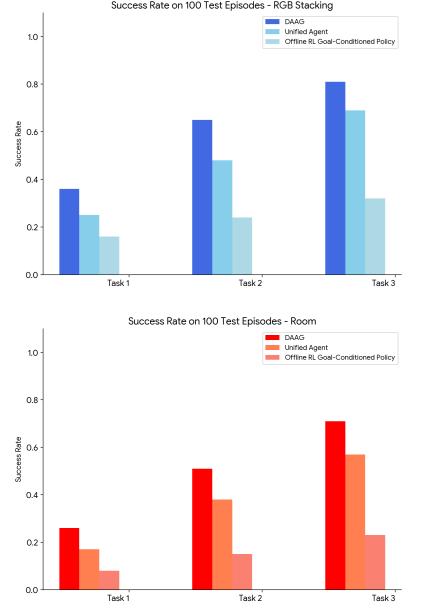


Figure 9: Sequential task learning performance. By learning to repurpose also episodes solving different but related tasks via HEA, DAAG shows superior transfer learning and lifelong learning performance.

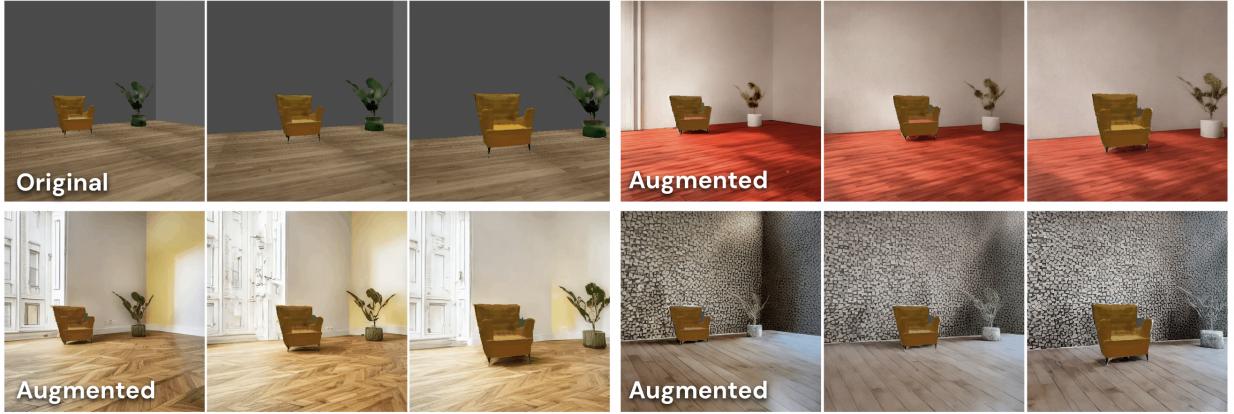


Figure 10: Examples of original observations from the Room environment and augmentations obtained through our geometrically and temporally consistent diffusion pipeline.

#### 4.5 IMPROVING ROBUSTNESS VIA SCENE VISUAL AUGMENTATION

We demonstrate how our diffusion pipeline can also be used to visually augment visual observations by modifying the scene and keeping the salient objects untouched, therefore generating additional examples of successful trajectories with different backgrounds. This is possible thanks to both the geometrical consistency and temporal consistency, absent in methods like Chen et al. (2023); Mandi et al. (2023); Yu et al. (2023a). To test how this affects policy robustness, we gather a dataset of 300 successful episodes in the Room environment where an agent reaches the yellow chair.

We then use our pipeline to augment each observation 5 times, querying the LLM to propose a description of an augmentation (e.g. *a room with a red floor and white walls*). We add all these augmented observations to our buffer and train a policy on it. Both the policy trained on the original and augmented datasets are tested on 5 visually modified room, where we randomly change the walls and floor colors as well as the distractor objects, running 20 test episodes on each room. Figure 11 demonstrated how visual augmentations leads to a substantially more robust policy, able to reach the target object also on rooms that appear very visually different from the single training room.

## 5 CONCLUSION

In this work, we proposed Diffusion Augmented Agent (DAAG), a framework that combines large language models, vision-language models, and diffusion models to tackle key challenges in lifelong reinforcement learning for embodied AI agents. Specifically, our key results show that DAAG can accurately detect rewards on novel, unseen tasks where traditional approaches fail to generalize. By repurposing experience from prior tasks, DAAG progressively learns each subsequent task more efficiently, requiring fewer episodes thanks to transfer learning. Finally, by diffusing unsuccessful episodes into successful trajectories for related subgoals, DAAG substantially improves exploration efficiency. Through diffusion augmentation, experience gathered across a lifetime of learning can be repurposed to make each new task easier than the last. This work suggests promising directions for overcoming data scarcity in robot learning and developing more generally capable agents.

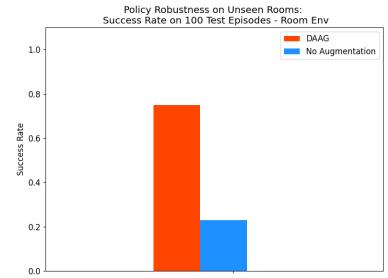


Figure 11: Success rates of baseline policy and a policy trained on augmented dataset (10).

## 6 ACKNOWLEDGMENTS

The authors would like to thank Dushyant Rao for his valuable feedback on earlier drafts of the paper.

## REFERENCES

- M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022.
- J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan. Flamingo: a visual language model for few-shot learning, 2022.
- M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.
- A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, V. Jampani, and R. Rombach. Stable video diffusion: Scaling latent video diffusion models to large datasets, 2023.
- R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang. On the opportunities and risks of foundation models, 2022.
- K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauza, T. Davchev, Y. Zhou, A. Gupta, A. Raju, A. Laurens, C. Fantacci, V. Dalibard, M. Zambelli, M. Martins, R. Pevceviciute, M. Blokzijl, M. Denil, N. Batchelor, T. Lampe, E. Parisotto, K. Žohna, S. Reed, S. G. Colmenarejo, J. Scholz, A. Abdolmaleki, O. Groth, J.-B. Regli, O. Sushkov, T. Rothörl, J. E. Chen, Y. Aytar, D. Barker, J. Ortiz, M. Riedmiller, J. T. Springenberg, R. Hadsell, F. Nori, and N. Heess. Robocat: A self-improving generalist agent for robotic manipulation, 2023.
- A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.
- T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020.
- S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. de Freitas, and Z. Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning, 2020.
- L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

- Z. Chen, S. Kiami, A. Gupta, and V. Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation, 2023.
- C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2023.
- O. X.-E. Collaboration. Open x-embodiment: Robotic learning datasets and rt-x models, 2023.
- P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis, 2021.
- N. Di Palo, A. Byravan, L. Hasenclever, M. Wulfmeier, N. Heess, and M. Riedmiller. Towards a unified agent with foundation models, 2023.
- A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems, 2021.
- R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, et al. Foundation models in robotics: Applications, challenges, and the future. *arXiv preprint arXiv:2312.07843*, 2023.
- Gemini-Team. Gemini: A family of highly capable multimodal models, 2023.
- D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai. Retinagan: An object-aware approach to sim-to-real transfer, 2021.
- J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020.
- J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre. Training compute-optimal large language models, 2022.
- W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022.
- W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models, 2023.
- L. Khachatryan, A. Movsisyan, V. Tadevosyan, R. Henschel, Z. Wang, S. Navasardyan, and H. Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators, 2023.
- I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning, 2021.
- T. Kwon, N. D. Palo, and E. Johns. Language models as zero-shot trajectory generators, 2023.
- A. X. Lee, C. M. Devin, Y. Zhou, T. Lampe, K. Bousmalis, J. T. Springenberg, A. Byravan, A. Abdolmaleki, N. Gileadi, D. Khosid, et al. Beyond pick-and-place: Tackling robotic stacking of diverse shapes. In *Conference on Robot Learning*, pages 1089–1131. PMLR, 2022.
- J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control, 2023.
- C. Lu, P. J. Ball, Y. W. Teh, and J. Parker-Holder. Synthetic experience replay, 2023.
- C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time, 2022.
- Z. Mandi, H. Bharadhwaj, V. Moens, S. Song, A. Rajeswaran, and V. Kumar. Cacti: A framework for scalable multi-task multi-scene visual imitation learning, 2023.
- V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning, 2013.
- J. Oh, Y. Guo, S. Singh, and H. Lee. Self-imitation learning. In *International Conference on Machine Learning*, pages 3878–3887. PMLR, 2018.
- OpenAI. Gpt-4 technical report, 2023.

- A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision, 2021.
- A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation, 2021.
- R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020.
- S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent, 2022.
- R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.
- D. I. A. Team, J. Abramson, A. Ahuja, A. Brussee, F. Carnevale, M. Cassin, F. Fischer, P. Georgiev, A. Goldin, M. Gupta, T. Harley, F. Hill, P. C. Humphreys, A. Hung, J. Landon, T. Lillicrap, H. Merzic, A. Muldal, A. Santoro, G. Scully, T. von Glehn, G. Wayne, N. Wong, C. Yan, and R. Zhu. Creating multimodal interactive agents with imitation and self-supervised learning, 2022.
- D. Tirumala, T. Lampe, J. E. Chen, T. Haarnoja, S. Huang, G. Lever, B. Moran, T. Hertweck, L. Hasenclever, M. Riedmiller, N. Heess, and M. Wulfmeier. Replay across experiments: A natural extension of off-policy rl, 2023.
- J. Wang, Z. Wu, Y. Li, H. Jiang, P. Shu, E. Shi, H. Hu, C. Ma, Y. Liu, X. Wang, et al. Large language models for robotics: Opportunities, challenges, and perspectives. *arXiv preprint arXiv:2401.04334*, 2024.
- L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023.
- T. Xiao, H. Chan, P. Sermanet, A. Wahid, A. Brohan, K. Hausman, S. Levine, and J. Tompson. Robotic skill acquisition via instruction augmentation with vision-language models, 2023.
- T. Yu, T. Xiao, A. Stone, J. Tompson, A. Brohan, S. Wang, J. Singh, C. Tan, J. Peralta, B. Ichter, et al. Scaling robot learning with semantically imagined experience. *arXiv preprint arXiv:2302.11550*, 2023a.
- W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia. Language to rewards for robotic skill synthesis, 2023b.
- L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models, 2023.
- J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.
- Z. Zhu, H. Zhao, H. He, Y. Zhong, S. Zhang, H. Guo, T. Chen, and W. Zhang. Diffusion models for reinforcement learning: A survey, 2024.

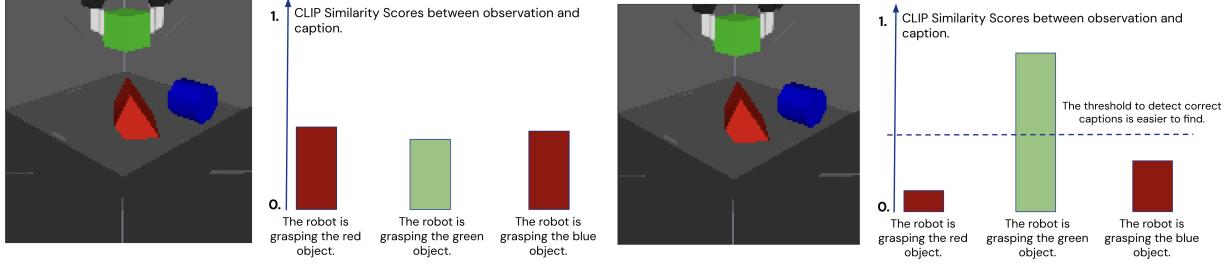


Figure 12: Illustrative comparison between the outputs of CLIP before and after finetuning, and how finetuning makes it possible to find a threshold to discriminate correct and wrong text-image matches.

## 7 APPENDIX

### 7.1 BACKWARD TRANSFER WITH HINDSIGHT EXPERIENCE AUGMENTATION

In the experiments of section 4.4 we studied forward transfer in a lifelong learning setting. We also analysed backward transfer in the RGB Stacking task as follows. After each task of Figure 9, we also test performance on the previous tasks by running HEA on all observations gathered up to that points to synthesise additional examples for previous tasks, and retrain and test the policy. Figure 13 demonstrates how HEA unlocks also strong backward transfer on the same three tasks order as Fig. 9: as the agent receives new data for new tasks, HEA can augment this data also to improve performance on previous tasks if needed.

### 7.2 PERFORMANCE WITH PURELY RANDOM EXPLORATION ON RGB STACKING

In our main experiments we used guided exploration to speed up the experiments. However, guided exploration does not change the relative performance of the various methods. We demonstrate how DAAG learns tasks faster than the chosen baseline by re-running the experiment in 8 (top), using entirely random exploration, where the agent samples a random position to pick and place the objects.

Figure 14 demonstrates how DAAG can learn to tackle a task considerably faster than the baseline that does not use HEA also in this scenario.

### 7.3 FINETUNING CLIP AND FINDING A THRESHOLD

As a contrastive VLM, CLIP outputs a similarity score between a textual description and an image. In our work, as also found in (Di Palo et al., 2023), we observed that off-the-shelf CLIP struggles at recognising precise configurations of objects, performing close to random guessing. We therefore finetune it as described in the main paper, increasing the difference in score between correct text-image matches and wrong matches. Given the dataset we used for finetuning, we then find  $\delta$ , the threshold to detect if a match is correct by comparing if the similarity score is larger, simply by finding the value that would maximise accuracy in an held-out validation set taken from the training set. In Figure 12 we illustrate this behaviour.

### 7.4 COMPARING DIFFERENT AUGMENTATION TECHNIQUES

In this work, we proposed a diffusion pipeline to visually modify and augment observations, focusing on geometrical and temporal consistency. Here we compare our pipeline with another technique from the recent literature, a CycleGAN-based pipeline Zhu et al. (2020) inspired by RetinaGAN Ho et al. (2021). We compare our technique and a RetinaGAN-like technique on the RGB Stacking

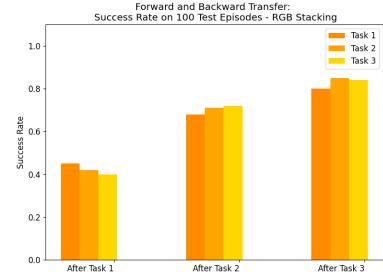


Figure 13: Backward transfer in a lifelong learning scenario using HEA.

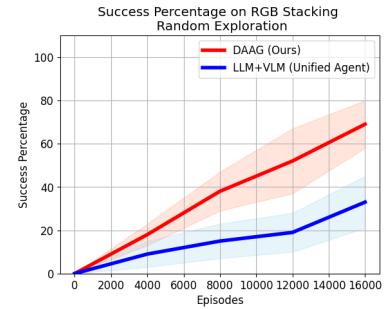


Figure 14: Performance of DAAG against the baseline with purely random exploration.

environment. In particular, we collect 300 successful episodes for the task "Stack Blue on Green". We then use both techniques to augment this dataset to a new task, "Stack Red on Blue". We compare the performance when receiving 0, 50, 100 or 200 successful examples of the new task. While we can use off-the-shelf Diffusion Models, trained on web-scale datasets, we need to train the CycleGAN on the data from the new task and old task in order to learn a visual mapping. Due to this, DAAG achieves strong 0-shot forward transfer, and keeps improving, while the RetinaGAN-like baseline needs 100 successful examples of the new task to properly learn a visual mapping and re-use data from the old task for the new one, as can be seen in Figure 15.

## 7.5 DESIGN OF LLM PROMPT

In this work, the LLM has two main roles: dividing tasks into sub-goals, and comparing tasks to detect if one can be visually modified into the other. For the former, we follow the same prompt proposed in (Di Palo et al., 2023). For the latter, we illustrate the prompts we used in Figure 16. We use a two stages approach: first, the LLM proposes a possible swap in a format that is easy to parse. Then we apply the swap to the task strings, and compare it again to reduce possible errors.

## 7.6 MODELS

### USED TO COMPUTE THE INPUTS TO CONTROLNET

As visual inputs to ControlNet, we provide, given the original RGB observation, 1) a *canny edges* image that is obtained via OpenCV 2) a *depth image* that we compute using the off-the-shelf model MiDaS (Ranftl et al., 2020) 3) a normals map that is computed algorithmically from the aforementioned depth map.

All these inputs guide the generation process, and we use a weight of 0.8 for each in the ControlNet pipeline, using the Diffusers library <https://huggingface.co/docs/diffusers/en/index>.

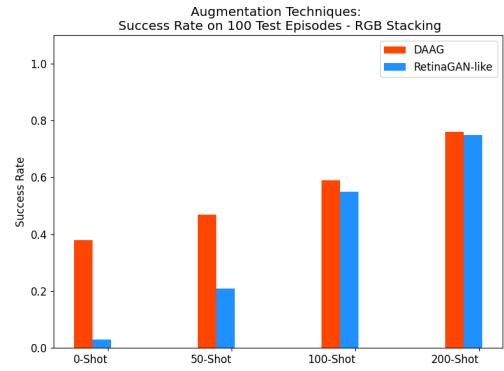


Figure 15: Performance of DAAG against a RetinaGAN-like style augmentation technique to learn a new task given successful examples of a different tasks and varying amounts of successful examples of the new task at hand.

Help me compare these pairs of tasks I give you, telling me if you can swap an object from task 1 to get task 2. You can only swap objects that are similar in shape! Your response should be "YES/NO - {Object 1} - {Object 2} - {Small Explanation}" and nothing else.

EXAMPLES:

Task 1: The robot is grasping the red cube.  
Task 2: The robot is grasping the blue cube.

ANSWER:

YES - red cube - blue cube - a red and a blue cube are similar objects

Task 1: The robot is grasping the red cube.  
Task 2: The robot is grasping the hammer.

ANSWER:

NO - a red cube and an hammer are not similar objects

Task 1: The robot is grasping the red cube.  
Task 2: The blue cube is on top of the green cube.

ANSWER:

NO - the two tasks are different

Task 1: The apple is on the blue chair.  
Task 2: The computer is on the blue chair.

ANSWER:

NO - an apple and a computer have too different shapes

Help me compare these pairs of tasks I give you, telling me if they represent the same task. Your response should be "YES/NO" and nothing else.

EXAMPLES:

Task 1: The robot is grasping the red cube.  
Task 2: The robot is grasping the blue cube.

ANSWER:

NO

Task 1: The robot is grasping the red cube.

Task 2: The robot is grasping the red cube.

ANSWER:

YES

Task 1: The robot is grasping the red cube.

Task 2: The blue cube is on top of the green cube.

ANSWER:

NO

Task 1: The apple is on the gray chair.

Task 2: The lemon is on the blue chair.

ANSWER:

NO

Task 1: The apple is on the gray chair.

Task 2: The apple is on the blue chair.

ANSWER:

NO

Figure 16: Prompts used to detect if a task can be visually modified into another by swapping objects.

## 7.7 MODELS AND HYPERPARAMETERS

**Large Language Model:** Gemini Pro 1.0

**Vision Language Model:** CLIP ViT-B/32

**Diffusion Model:** Stable Diffusion 1.5 with ControlNets (Canny, Depth, Normals)

*Hyperparameters:* Image generation size:  $512 \times 512$ . Weights: [0.5, 0.8, 0.8], Generation steps: 20, Guidance Scale = 7

**Policy Network:** ResNet-18 + 2-layer MLP

**Super Resolution Model:** Stable Diffusion 4x Upscaler

**Object Detection Model:** OWL-ViT

**Segmentation Model:** FastSAM + CLIPSeg

**Reinforcement Learning Hyperparameters:**  $\epsilon = 0.99$ , decay: 0.9995 per episode

*Policy Training:* batch size 32, optimiser Adam, learning rate  $1e - 4$ , epochs 5000.