Extended Abstract

Motivation Vision-based robotic control has emerged as a critical capability for autonomous systems operating in complex, unstructured environments. The ability to directly process raw visual observations and generate appropriate control actions represents a significant step toward more generalizable and adaptable robotic systems. However, finding an optimal policy often requires multi-step reasoning over long-horizon tasks and extracting feedback in environments with sparse reward signals. Vision-based robotic manipulation has proven challenging for conventional reinforcement-learning (RL) pipelines because reward signals are often sparse, and the state-action space is exceptionally high-dimensional.

Method We introduce Dreamer Policy, a hierarchical framework that separates long-horizon planning from low-level control. We have a high-level conditional diffusion model generate feasible future frames given a current image observation state. These proposed subgoals are then given to the goal-conditioned low-level policy, which uses Proximal Policy Optimization (PPO) and Hindsight Experience Replay (HER) to learn how to reach the imagined state. We also densify the reward environment, originally a sparse binary 0/+1 environment, with a pixel-difference reward function.

Implementation We implement the visual predictor as a conditional UNet DDPM trained offline on 3,800 expert frame pairs drawn from the D4RL Minari kitchen complete-v2 dataset. The network denoises 120×120 grayscale images across 1,000 diffusion steps, leveraging sinusoidal time embeddigns, residual-attention blocks, and group normalization; training with Apple-Silicon MPS acceleration completes in roughly 100 epochs. The downstream control policy employs Proximal Policy Optimization (PPO) with observation flattening and vector normalization to ingest the concatenated robot state and 14,400-pixel goal representation.

Results Our Dreamer Policy obtains an average reward of 0.9669 after 85k timesteps, while our baseline PPO agent also is able to achieve 0.9669 average reward but after 94k timesteps. We attribute the 10% improvement to denser rewards due to our informative subgoals from the diffusion model. Qualitatively, we find that the agent following the Dreamer Policy is able to achieve high level subgoals at earlier timesteps (ex. grasping the kettle) for the kettle moving subtask. These findings highlight two contributions:

- **Generative Visual Foresight for RL**. We demonstrate that off-policy diffusion models can supply semantically rich sub-goals that accelerate exploration without any task-specific reward engineering.
- **Dense Reward via Pixel Space**. A length-normalized L2 distance between predicted and observed frames yields a universally applicable shaping signal that preserves sparse-reward correctness while vastly improving gradient frequency.

Discussion Our approach was largely limited by the binary and sparse nature of the FrankaKitchen environment, as vision-only agents were unable to learn long-horizon tasks without meaningful signals. To combat this, we incorporated the original proprioceptive observation space, but this undermines a deployable vision-only solution, which may be required in real-world scenarios. We faced compute bottlenecks with OpenGL rendering, which significantly reduced our training speed. Another major constraint was our limited access to larger-scale compute. Training vision-conditioned agents and diffusion models, especially with high-dimensional image inputs is extremely computationally intensive. As a result, we were forced to scale down our model to make up for this, and iteration loops took extremely long.

Conclusion In sum, Dreamer Policy illustrates a scalable path toward unifying modern generative modeling with continuous-control RL. Future work will assess purely vision-conditioned variants (no privileged proprioception), extend evaluation to contact-rich assembly domains, and explore tighter architectural fusion–e.g., transformer planners that share embeddings between the denoiser and the policy network.

A Dream Is All You Need

Devin Gupta, Ali Ahmad, Annie Lee
Department of Computer Science
Stanford University
{devgupta, aliahmad, annieee}@stanford.edu

Abstract

Robotic manipulation in visually rich, sparse-reward environments remains sample-inefficient for standard reinforcement-learning (RL) methods because the agent must first discover salient contact events before it can learn to exploit them. We propose Dreamer Policy, a model-based auxiliary that pairs a conditional diffusion model with a proximal-policy-optimization (PP0) backbone: at every timestep the diffusion model predicts an image of a desirable future state given the current observation, and the policy receives both the raw proprioceptive state and this predicted frame. The resulting dense, pixel-space gradients serve simultaneously as shaped rewards and as informative sub-goals, guiding exploration toward semantically meaningful object interactions. Our results demonstrate that generative visual foresight can substantially accelerate policy learning without task-specific reward engineering, suggesting a practical avenue for integrating modern diffusion models with deep RL in real-world manipulation settings.

1 Introduction

Vision-based robotic control has emerged as a critical capability for autonomous systems operating in complex, unstructured environments. The ability to directly process raw visual observations and generate appropriate control actions represents a significant step toward more generalizable and adaptable robotic systems. However, finding an optimal policy often requires multi-step reasoning over long-horizon tasks and extracting feedback in environments with sparse reward signals.

Hierarchical reinforcement learning (HRL) Barto and Mahadevan (2003) offers a solution to these problems by decomposing a long-horizon task into a high-level policy that dictates sub-goals and a low-level policy that executes actions to reach these sub-goals. This breakdown provides an advantage during both training and exploration. The high-level policy corresponds to multiple low-level environment steps, so we can propagate reward across a few high-level choices which speeds up learning. Since exploration occurs at a higher level, the agent learns more meaningful trajectories based on abstracted goals instead of atomic action sequences.

Meanwhile, denoising diffusion probablistic models (DDPMs), which gradually adds and removes random noise from structured data, have shown success with synthetic data generation. Taking advantage of their generative capabilites, methods like Diffusion Policy (Chi et al., 2023) utilizes this mechanism to reframe action sequence generation as a denoising problem.

We offer Dreamer Policy, a bridge between these paradigms and present a hierarchical framework that combines a diffusion policy and a low-level action policy. Our key objectives are sub-goal generation, goal-conditioned control, and reward densification. First, our conditional diffusion model iteratively denoises the current image, learning to imagine feasible future states from the current visual observations. Then, our Soft Actor-Critic (SAC) policy, augmented with Hindsight Experience Replay (HER), takes in the current state and imagined future state from the diffusion model and learns to execute sequences of actions to reach that goal. We introduce a simple pixel-difference reward,

scaling the Euclidean distance between the current and goal image, to give the agent a continuous reward signal at each timestep instead of the default binary sparse feedback.

With our architecture, we ask a few core questions. Can offline diffusion produce high quality sub-goals and what factors most influence the accuracy of the denoised frame compared to the true next observation? How can we design the low-level policy to reliably track the diffusion-proposed imagined subgoals in our long-horizon tasks? How effective is pixel-difference reward densification in improving performance compared to the default reward signal?

We evaluate our method on the Frank Kitchen environment, which presents the difficulties aforementioned: long-horizon planning across the 4 tasks and sparse reward signals (+1 only when a task is fully completed, 0 otherwise). Our experiments demonstrate that our Dreamer Policy outperforms the state-only Proximal Policy Optimization (PPO) and SAC baselines for both sample efficiency and learning speed, confirming our hypothesis that imagined visual sub-goals can meaningfully densify sparse rewards and guide exploration.

2 Related Work

Hierarchical Planning and Control: Early off-policy HRL methods such as HIRO (Nachum et al., 2018) use a neural network policy for their high-level controller to directly output subgoals which exist in the same state space as the environment. The low-level goal-conditioned policy learns to reach the proposed goals and requires off-policy correction. Later frameworks like HILL (Yang et al., 2021) first learn a model of reachability like a graph-based representation of the environment dynamics to pick feasible intermediate goals. Our work builds off of HIRO's hierarchical framework by using a diffusion model to imagine future images and working with visual observations instead of state vectors, keeping vision-based robotic control in mind. We take the core idea of learning a reachable space to propose subgoals from HILL and later works but work with image pixels instead of an embedding space.

Diffusion Models for Motion Planning Denoising diffusion probabilistic models recast traditional motion planning (finding trajectories by sampling over state spaces) as a generative sampling problem. DDPMs first had their breakthrough impact on high quality image synthesis by learning to gradually denoise for image recovery. More recently, diffusion models have been used for planning robot actions. Diffuser (Janner et al., 2022) treats action plan as random noise and iteratively refines it to be a feasible trajectory. Diffusion Policy (Chi et al., 2023) treats the policy as a diffusion process over multi-step action rollouts, denoising the action sequences to obtain a multi-step motion plan. While this is frames it as online planning, we conduct motion planning by using diffusion as a offline subgoal generator.

Dense Reward Learning with HER Sparse rewards pose a major challenge for systems to learn how to reach long-term goals, as agents will most often receive no reward and thus no informative feedback about their actions. Since it's difficult to learn by pure trial and error in a sparse reward environment, Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) takes advantage of all the failed states by relabeling the fianl state we reached in a trajectory as a success, pretending the reached state was the intended goal. This creates more useful +1 signals that guide the agent to learn from what it did achieve rather than only what it's supposed to achieve. In our work, we combine HER's relabeling idea with diffusion generated visual goals and a pixel-distance metric (scaled Euclidean difference between the current image and the diffusion dreamed state). This provides us with a dense learning signal even though our kitchen environment's native reward is binary.

3 Method

Our approach combines diffusion-based visual prediction with reinforcement learning in a self-play framework for robotic manipulation tasks in the D4RL Kitchen environment. The methodology consists of three main components: a conditional diffusion model for predicting future visual states, a custom environment wrapper that integrates diffusion predictions into the observation structure, and reinforcement learning agents trained using Proximal Policy Optimization (PPO). This integration enables agents to learn goal-directed behavior by pursuing visual states predicted by the learned

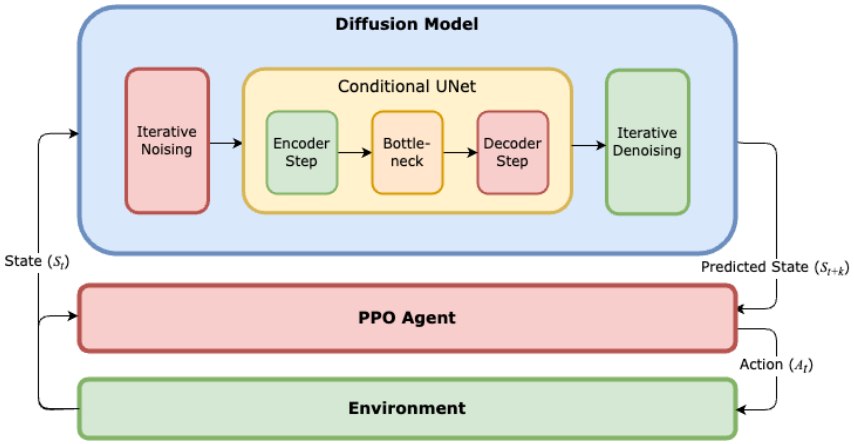

Figure 1: The Dreamer Policy architecture. The environment passes an image of its current state to the soft actor-critic (SAC) agent and diffusion model. The diffusion model predicts the next state which then passes it through to the agent, which acts on the environment, causing a loop. The diffusion model acts as a planner to the SAC Agent in a hierarchical manner.

world model, creating a self-supervised learning paradigm where the agent's goals emerge from its understanding of environmental dynamics.

3.1 Environment and Data Processing

3.1.1 Kitchen Environment Setup

We utilize the D4RL Kitchen environment Fu et al. (2020) through the Minari framework Younis et al. (2024), specifically employing the `FrankaKitchen-v1` environment which features a 7-DOF Franka Panda robotic arm performing various kitchen manipulation tasks including kettle operation, microwave control, and cabinet manipulation. The environment provides a continuous action space of dimension 9, comprising 7 joint velocities and 2 gripper actions, with RGB observations rendered at 480×480 resolution. Task-specific goal configurations enable completion detection, while the episodic structure supports variable-length trajectories of up to 280 steps, providing sufficient temporal horizon for complex manipulation sequences.

3.1.2 Dataset Construction

The data processing pipeline creates frame pairs for training the diffusion model through the following steps:

1. **Episode Selection**: We process episodes from the `D4RL/kitchen/complete-v2` dataset, selecting the first 5 episodes per dataset batch to balance computational efficiency with diversity.
2. **Frame Pair Generation**: For each episode, we create pairs (I_t, I_{t+k}) where I_t is the current frame and I_{t+k} is the frame k steps into the future. The default value is $k = 5$ timesteps.
3. **Episode Truncation**: Episodes are truncated at the last successful task completion to ensure meaningful state transitions, preventing the model from learning from post-completion drift.
4. **Image Preprocessing**:
   - Resize from $480 \times 480 \times 3$ to $120 \times 120 \times 1$
   - Convert RGB to grayscale for computational efficiency
   - Normalize pixel values to $[-1, 1]$ range
   - Store as pre-processed tensors in CHW format

The resulting dataset contains approximately 3,809 frame pairs for the default configuration, cached in pickle format for efficient loading in subsequent training runs.

3.2 Diffusion Model Architecture

3.2.1 Model Design

We implement a conditional U-Net architecture following the Denoising Diffusion Probabilistic Model (DDPM) framework Ho et al. (2020), where the model learns to predict the noise $\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})$ added to future frames conditioned on current frame observations. The input configuration consists of the current frame $\mathbf{c} \in \mathbb{R}^{1\times120\times120}$ represented as a grayscale image, the noisy future frame $\mathbf{x}_t \in \mathbb{R}^{1\times H\times W}$, which are concatenated to form the input $[\mathbf{x}_t, \mathbf{c}] \in \mathbb{R}^{2\times120\times120}$. The temporal information is encoded through timestep embeddings $t \in \{1, 2, \ldots, T\}$ where $T = 1000$ represents the total number of diffusion steps.

3.2.2 Architecture Components

Time Embedding: Temporal conditioning employs sinusoidal positional encoding to transform discrete timesteps into dense vector representations:

$$\text{PE}(t, 2i) = \sin(t/10000^{2i/d}), \quad \text{PE}(t, 2i+1) = \cos(t/10000^{2i/d}) \tag{1}$$

where $d = 128$ is the embedding dimension.

U-Net Structure: The U-Net architecture follows a hierarchical encoder-decoder structure. The encoder implements a downsampling path with channel multipliers $(1, 2, 3)$. The bottleneck layer incorporates residual blocks with self-attention at the lowest resolution. The decoder employs an upsampling path with skip connections from corresponding encoder layers. The base channel count of 32 undergoes progressive multiplication at each hierarchical level.

Residual Blocks: Residual blocks incorporate group normalization with 32 groups, SiLU activation functions, and time embedding injection through learned linear projections. Dropout regularization uses a rate of 0.1, with skip connections for gradient flow.

Attention Mechanism: Self-attention applied at resolution 32×32 using multi-head attention (8 heads) to capture long-range spatial dependencies.

3.2.3 Training Procedure

The diffusion process follows the standard DDPM formulation with linear noise scheduling:

$$\beta_t = \beta_{\text{start}} + \frac{t-1}{T-1}(\beta_{\text{end}} - \beta_{\text{start}}) \tag{2}$$

where $\beta_{\text{start}} = 0.0001$ and $\beta_{\text{end}} = 0.02$.

3.2.4 Training Procedure

The diffusion process adheres to the standard DDPM formulation with linear noise scheduling, where the noise variance follows the schedule:

$$\beta_t = \beta_{\text{start}} + \frac{t-1}{T-1}(\beta_{\text{end}} - \beta_{\text{start}}) \tag{3}$$

with $\beta_{\text{start}} = 0.0001$ and $\beta_{\text{end}} = 0.02$.

The training algorithm samples frame pairs $(I_{\text{current}}, I_{\text{future}})$ from the dataset, timestep $t \sim \text{Uniform}(1, T)$, and Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Noisy images are computed as $I_t = \sqrt{\bar{\alpha}_t} I_{\text{future}} + \sqrt{1-\bar{\alpha}_t}\epsilon$, where $\bar{\alpha}_t$ is the cumulative noise schedule parameter. The U-Net predicts noise $\epsilon_\theta = \text{UNet}([I_t, I_{\text{current}}], t)$, with parameters updated via MSE loss $\mathcal{L} = \|\epsilon - \epsilon_\theta\|_2^2$ using AdamW optimizer with learning rate 10^{-4}.

Training uses Apple Silicon MPS acceleration with batch size 2 over 100 epochs. Memory optimization includes gradient checkpointing and mixed precision when available.

3.3 Reward Function Design

In our custom environment, we had an optional flag for a custom reward function based on a pixel difference. We weren't able to explore alternatives to a L2 norm approach, although mentors suggested SSIM or alternative representations that are more information-dense in image space.

The reward function architecture promotes convergence toward visually coherent manipulation goals through a carefully designed similarity metric. The reward formulation:

$$r(s_t, a_t, s_{t+1}) = -\|\text{achieved\_goal}_{t+1} - \text{desired\_goal}_t\|_2 \tag{4}$$

encourages the agent to minimize the L2 norm between achieved and desired visual states, providing dense feedback based on visual similarity rather than relying on sparse task completion signals. This dense reward structure enables more efficient learning by offering continuous guidance toward the predicted goal states, facilitating smoother policy optimization and reducing the exploration burden typically associated with sparse reward environments.

3.4 Reinforcement Learning Agents

3.4.1 Proximal Policy Optimization (PPO) Implementation

We employ Proximal Policy Optimization (PPO) Schulman et al. (2017) as our primary reinforcement learning algorithm, leveraging its demonstrated stability and sample efficiency characteristics with continuous action spaces. The algorithm's clipped objective function provides robust policy updates while preventing destructive policy changes, making it particularly well-suited for high-dimensional visual observation spaces and continuous control tasks.

The PPO implementation operates with a learning rate of 3×10^{-4}, discount factor $\gamma = 0.99$, and GAE parameter $\lambda = 0.95$ for advantage estimation. Training occurs over minibatches of size 64 sampled from collected rollouts of 2048 timesteps, with policy updates performed over 10 optimization epochs per rollout. The clip range is set to 0.2 to constrain policy updates and maintain training stability, while entropy regularization with coefficient 0.01 encourages exploration during policy learning.

The actor-critic architecture employs fully connected networks with shared features for both policy and value functions. The network architecture consists of two hidden layers of 256 neurons each with ReLU activations, followed by separate heads for policy output (actor) and value estimation (critic). Environment normalization ensures training stability through `VecNormalize`, which applies standardization according to:

$$\hat{o}_t = \frac{o_t - \mu_o}{\sqrt{\sigma_o^2 + \epsilon}} \tag{5}$$

where μ_o and σ_o^2 represent running estimates of observation mean and variance, computed incrementally during training to adapt to the evolving observation distribution.

3.4.2 Observation Space Processing

The reinforcement learning agent operates on a custom flattened observation space that integrates both environmental state information and visual predictions from the diffusion model. The observation wrapper (`FlattenObservationWrapper`) transforms the original dictionary-based observation space into a unified vector representation comprising robot state variables, goal configurations, and processed visual information.

Visual processing converts RGB observations to 120×120 grayscale images, which are then processed through either the current frame (baseline condition) or diffusion-predicted goal images (experimental condition). The processed images undergo normalization to the $[-1, 1]$ range and are flattened into the observation vector alongside proprioceptive state information. This unified representation enables the policy network to simultaneously reason about spatial relationships, manipulation objectives, and visual goal configurations within a single neural architecture.

4 Experimental Setup

4.1 Data / Task Description

We use the `FrankaKitchen-v1` environment from the `gymnasium_robotics` suite, introduced in Gupta et al. (2019). The environment simulates a 9 degree of freedom (DOF) Franka Emika Panda robotic arm operating in a physically modeled kitchen. The base environment contains many tasks, however we primarily operate on two tasks: replacing kettle on a different burner and opening a microwave door. We selected this environment for three key reasons:

1. **Sparse reward signal.** The base environment only provides a $+1$ reward upon subtask completion (defined by an $L_2 < 0.3$ for relevant object configuration) and a -0 otherwise. In longer horizon, high dimensional settings, this reward sparsity proves to be a particularly challenging task. As a result, we expect a successful visual reinforcement to perform better than the base policy.
2. **High-dimensional state and action space**: The observation space is 59-dimensional, incorporating the robot's joint positions and velocities along with the poses and velocities of all task-relevant kitchen objects. The action space is a 9-dimensional continuous control space, where each dimension corresponds to either a joint angular velocity (for the 7 arm joints) or linear velocity (for the 2 gripper fingers). The complexity of the action space makes the environment well-suited for evaluating the efficacy of visual policy learning under real-world robotics constraints.
3. **Transferability of Image-based Goal Representations**: Because the environment exposes image-based representations of the observation (generally for human-rendering), this problem provides a nicely packaged mechanism of confronting a challenge related to simulation to real transfer. Namely because agents must infer task progress from 2D projections of a 3D scene, they have a fundamentally underspecified problem (consider occlusions and depth [1]). Partial observability makes goal inference and reward specification especially challenging.

In all experiments, we use a four-task configuration for multitask evaluation, setting `tasks_to_complete = [‘microwave", ‘kettle", ‘bottom burner", ‘light switch"]`. Each task corresponds to a unique subset of object goal configurations. The robot always begins from a fixed initial configuration with all objects in their inactive or closed state and the kettle located on the bottom-left burner. We also don't include stochastic noise in the dynamics or action inputs.

Expert Dataset: Minari D4RL Kitchen Mixed-v2. Fortuantely, we were able to leverage a publicly available expert trajectory dataset provided via the `Minari` framework de Lazcano (2024). The dataset, `D4RL/kitchen/mixed-v2`, consists of 621 episodes and a total of $156{,}560$ time steps collected in the `FrankaKitchen-v1` environment.

- **Observation space:** Each timestep includes a dictionary with `observation` ($\in \mathbb{R}^{59}$), `achieved_goal`, and `desired_goal`. The `achieved_goal` and `desired_goal` fields provide low-dimensional task-specific representations:
  - `kettle`: 7D (free joint pose)
  - `bottom burner, light switch`: 2D (sliding joint values)
  - `microwave`: 1D (hinge joint angle)
- **Action space:** Continuous control space $\mathcal{A} = [-1, 1]^9$ with each dimension representing a velocity control signal for a specific robot joint (7 arm joints + 2 gripper fingers).
- **Episode length:** Each episode has a maximum length of 450 timesteps.

For our trained diffusion model, we learn to denoise $t + k$ states conditioned on t observation on this expert dataset. This provides some bootstrapping for goal learning and general long-horizon planning capabilities. Note for our low-level policy, we didn't warm-start our policies using this labeled dataset.

[1] In practice monocular depth estimation can/should be used to re-project depth information from single images, although in this work we don't use such techniques.

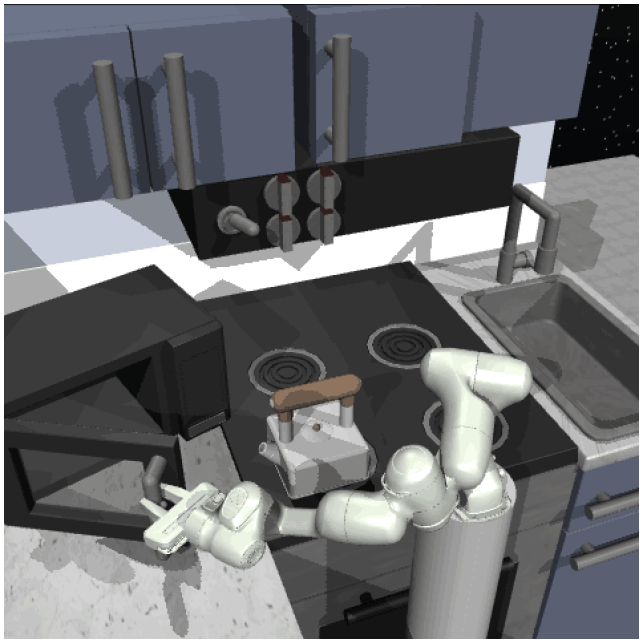

Figure 2: Successful expert trajectory with `'microwave'` subtasks in Minari D4RL Kitchen Mixed dataset. Many task sequences included four subtasks `'microwave'`, `'bottom burner'`, `'kettle'`, `'light switch'` in the same trajectory.

4.2 Evaluation Metrics and Baselines

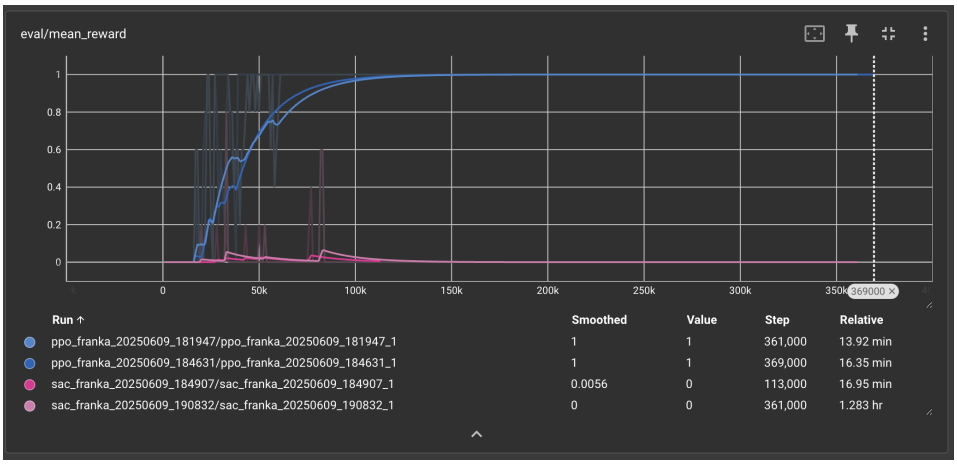

Figure 3: Proximal Policy Optimization significantly outperforms Soft-Actor Critic methods in the base environment for the `'kettle'` subtask within $350{,}000$ timesteps.

In order to get a baseline for our model, we applied our Soft-Actor Critic and Proximal Policy Optimization methods to the base environment with the 59 dimensional observation space. We ran our benchmark on the `'kettle'` subtask for ≈ 350k timesteps, 2 runs and saw significantly better performance on the PPO method over the SAC method. Specifically, the average reward for PPO is 1 while SAC is 0. There are two potential causes for this outperformance:

1. **On-policy learning**: Because PPO is an on-policy algorithm, because we have a large number of timesteps in these baselines, data-efficiency is considerably less of an advantage for SAC.

2. **Stability of Training**: The primary cause of this outperformance is due to stability of trianing. Because our action space is relatively large (9 dimensions), the entropy bonus terms in SAC's learned model are not efficient in learning a good state representation (especially given the sparse rewards).

5 Results

5.1 Quantitative Evaluation

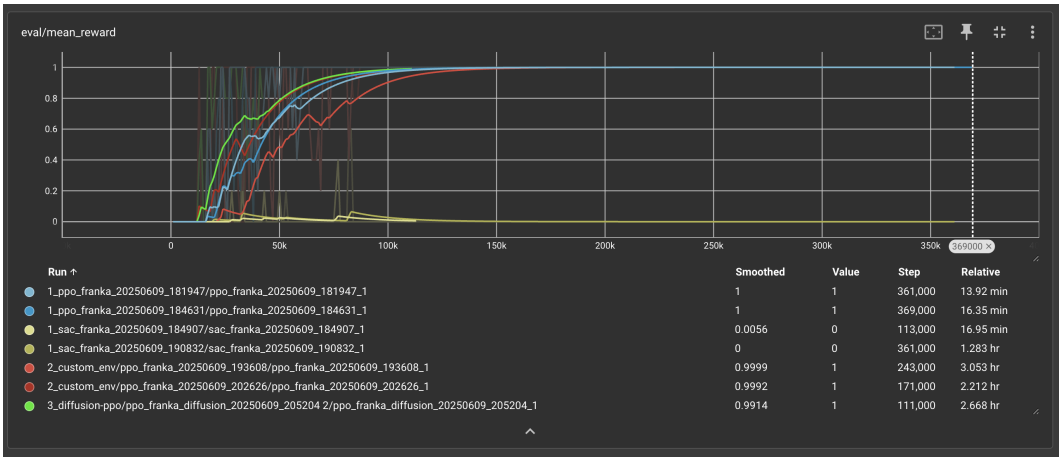

Figure 4: Comparison of various policy training strategies on the `kettle` task within the FrankaKitchen environment.

All results are set-seed. Unless otherwise noted, success is measured with the FrankaKitchen sparse-reward signal (reward is 1 on task completion, 0 otherwise). To isolate the contribution of diffusion-generated visual sub-goals we compare:

- **Dreamer Policy (ours)** - PPO agent conditioned on the predicted future frame and trained with dense pixel-difference reward.
- **State-only PPO** - identical network and hyper-parameters but without diffusion predictions
- **State-only SAC** - SAC agent using only state information and the sparse reward signal.

Figure 4's learning curves reveal that Dreamer Policy reaches 0.9669 evaluation mean reward after 85k timesteps, whereas state-only PPO requires 94k timesteps for 0.9669 evaluation mean. This indicates a 10 % improvement in sample efficiency attributable to denser rewards and informative sub-goals.

All experiments use fixed random seeds and evaluate success using the sparse reward signal provided by the FrankaKitchen environment, where a reward of 1 indicates task completion and 0 otherwise.

Baseline Comparison. We benchmarked Proximal Policy Optimization (PPO) and Soft-Actor Critic (SAC) on the original environment with a 59-dimensional observation space. The two PPO runs (light blue and cyan in Figure 4) rapidly converge to optimal performance (evaluation reward = 1.0) within approximately 90,000 steps. In contrast, SAC fails to learn altogether across two separate runs (dark and light yellow), plateauing at a mean reward of 0.0. This gap highlights PPO's robustness in sparse reward settings, especially with higher-dimensional action spaces.

Sample Efficiency and Training Stability. We observe that PPO consistently achieves success after fewer than 100,000 timesteps, with the fastest run reaching the maximum reward at 85,000 steps. SAC's instability, likely due to entropy scaling in a 9-dimensional action space, leads to noisy or degenerate exploration, and neither SAC run succeeds even after 350,000 timesteps.

Custom Environment and Visual Rewards. In our modified setting with pixel-based rewards and visual goal conditioning, several configurations were evaluated:

- **State-only PPO (blue)** achieved full task success (reward = 1) by 125k timesteps, indicating stable learning even under custom image-based reward signals.
- **State-plus Image Conditioning PPO (red)** converged slower, reaching 0.6 smoothed reward at 34,000 steps, showing instability likely due to variation in random seed or visual reward processing.
- **Diffusion-PPO (green)** achieved a 0.6892 smoothed reward by just 34,000 steps. Full task success was achieved by $\approx$ 90,000 steps.

These results suggest that diffusion-predicted sub-goals help bootstrap effective behaviors earlier in training. At 34,000 steps, diffusion-PPO leads all baselines, with over 0.68 smoothed reward compared to 0.42 for State-only PPO and near-zero for all SAC runs. This early advantage demonstrates improved exploration and credit assignment due to dense pixel-difference rewards derived from sub-goal predictions.

Overall, the inclusion of visual sub-goals improves sample efficiency and smooths early learning. SAC, while powerful in continuous control tasks with dense rewards, appears ill-suited to sparse visual control under our task setup.

5.2 Qualitative Analysis

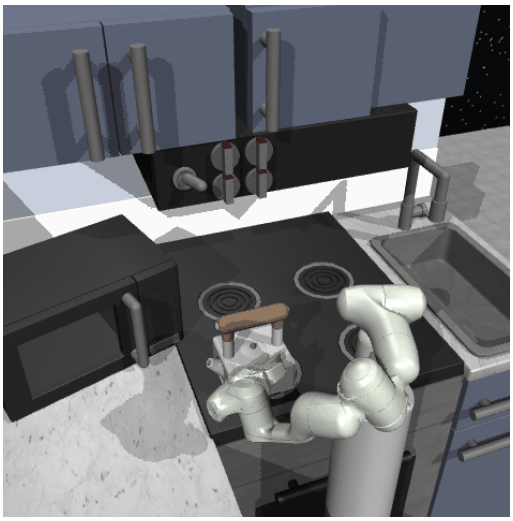

Figure 5: Qualitative results of the baseline policy on the custom environment.

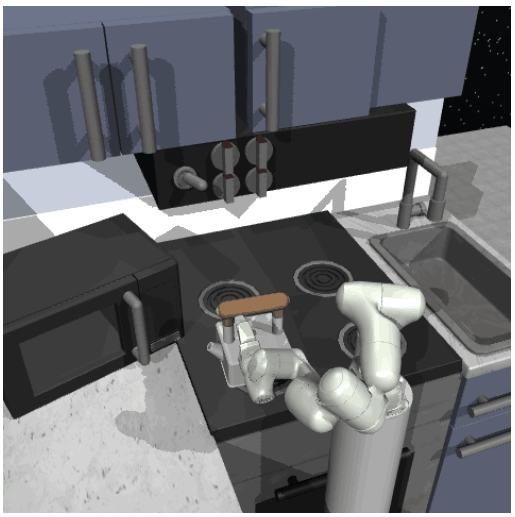

Figure 6: Qualitative results of the diffusion-conditioned policy.

Figure 7: Qualitative comparison of rolled out trajectories. The image on the left displays the trajectory from the baseline policy, while the image on the right shows the trajectory from the trained policy on diffusion conditioning. These static images are snapshots from animated GIFs, which represent the full rolled-out trajectories of the policies. Qualitatively, the diffusion-conditioned policy appears very similar to the baseline, though potentially exhibiting slightly less speed in its movements.

While generally both policies seemed to perform similarly, we do notice a couple of differences that showed up in the earlier rollout examples during the process of converging to optimal behavior. At the 50k training timestep, we recognize that the baseline agent struggles to grab the kettle at all, while the diffusion trained policy is able to grab the kettle but instead struggles to manipulate it coherently.

Firstly, we believe that the faster grasp acquisition of Dreamer Policy can be attributed to the affordance priors implicit in the predicted future frame. Because the diffusion model forecasts an image in which the kettle already rests in the gripper, the policy receives a dense pixel-wise gradient that pulls the end-effector toward visually salient grasp points. The baseline, guided only by sparse success/failure signals, must instead discover these key poses through undirected exploration; hence its slower progress.

Secondly, coherent post-grasp manipulation demands precise closed-loop control of joint angles and object orientation—information that is only indirectly encoded in the RGB prediction. In the absence of explicit torque or pose cues, Dreamer Policy initially oscillates between competing wrist trajectories, leading to the "jitter". The baseline, which learns purely from proprioceptive state transitions, exhibits less oscillation once a grasp is achieved, albeit at the cost of delayed grasp onset.

Thirdly, we observe a qualitative difference in exploration style. Diffusion-PPO trajectories are smoother and more goal-directed: the arm follows a single sweeping arc toward the kettle, whereas the baseline performs multiple aborted reaches. This suggests that imagined sub-goals not only guide the agent spatially but also regularize its action distribution, reducing high-frequency control noise.

6 Discussion

A primary limitation of our approach stemmed from the binary and sparse nature of the reward signal in the FrankaKitchen environment. In settings where the agent was trained solely with visual input—excluding low-dimensional state observations—learning failed to meaningfully progress. This challenge was exacerbated by the difficulty of credit assignment in long-horizon tasks with minimal supervision. In response, we incorporated the original proprioceptive observation space during training. However, this compromises the goal of deploying vision-only agents in real-world scenarios where such privileged information may not be available.

Another major constraint was our limited access to computational resources. Training vision-conditioned agents, particularly with high-dimensional image inputs and diffusion-based components, was computationally intensive. Due to dependencies on OpenGL-based rendering (essential for our image-based goal representations), we were unable to leverage headless VMs or traditional EC2 instances, which meant we lacked GPU-accelerated rendering capabilities. This significantly reduced our iteration speed, as we were restricted to on-device training with limited parallelism. Over the course of the project, multiple training runs required several days to complete, limiting the breadth of hyperparameter sweeps and averaging over runs we were able to study.

7 Conclusion

This work presents a promising direction for sample-efficient visual reinforcement learning through the use of diffusion-predicted subgoal conditioning. We demonstrate that imagined subgoal frames—generated via a conditional diffusion model—can accelerate task acquisition in long-horizon, sparse-reward environments such as FrankaKitchen. While the improvements over baselines remain modest under privileged settings, qualitative results suggest that diffusion-based goal conditioning can offer dense intermediate supervision in otherwise under-constrained tasks.

Future work can build on this framework in several meaningful directions:

1. Evaluate diffusion-conditioned policies in environments where no privileged state observations are available, and compare against agents conditioned solely on current-state image frames.
2. Benchmark against discrete-action baselines such as DQN to understand whether diffusion conditioning offers benefits beyond continuous control settings.
3. Investigate whether specifying a denser reward function—such as total final-state distance in pixel space—improves the performance of actor-critic methods.
4. Explore alternative architectures that fuse diffusion models and policies more tightly, such as transformer-based goal-predictive planning modules.

8 Team Contributions

- **Annie Lee:** Led preprocessing and development of the dataloader for the Minari dataset. Contributed to the diffusion model training scripts and collaborated on dataset handling, evaluation metrics, and the *Discussions/Future Work* section of the poster presentation.
- **Ali Ahmad:** Developed and refined diffusion model training pipelines. Integrated the trained diffusion model with the custom reinforcement learning environment and conducted final

policy evaluations. Also authored the *Methods/Experiments* and *Diffusion Model & Results* sections of the poster.

- **Devin Gupta:** Designed and implemented the custom reinforcement learning environment and authored training scripts compatible with Stable Baselines. Executed baseline comparisons and finalized the SAC agent. Contributed to the *RL Training* and *References* sections of the poster.

Changes from Proposal Our initial proposal centered on distillation from a Vision-Language-Action (VLA) model, with reinforcement learning via self-play serving as a task-specific fine-tuning phase. However, due to practical constraints—primarily compute limitations and the complexity of initializing a deployed VLA—we pivoted toward a more focused exploration of task-specific reinforcement learning, omitting reliance on a pretrained base model.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 30.

Andrew G. Barto and Sridhar Mahadevan. 2003. Recent Advances in Hierarchical Reinforcement Learning. *Discrete Event Dynamic Systems* 13, 1-2 (2003), 41–77.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. 2023. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. arXiv:2303.04137v5 [cs.RO] Preprint, arXiv:2303.04137v5.

Rodrigo de Lazcano. 2024. Minari: D4RL Kitchen Mixed-v2 Dataset. `https://github.com/rodrigodelazcano/d4rl-minari-dataset-generation`. Version 0.4.3.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. arXiv:2004.07219 [cs.LG]

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. 2019. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*. PMLR, 1025–1037.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. *arXiv preprint arxiv:2006.11239* (2020).

Michael Janner, Justin Fu, Ilya Sutskever, and Sergey Levine. 2022. Planning with Diffusion for Flexible Behavior Synthesis. In *Conference on Robot Learning (CoRL)*.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. 2018. Data-Efficient Hierarchical Reinforcement Learning. *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018).

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Qingkai Yang, Andrew Ho, Dieter Fox, Yoshua Yamaguchi, and Rahul Sukthankar. 2021. HILL: Hierarchical Imitation and Reinforcement Learning in Latent Space. In *International Conference on Robotics and Automation (ICRA)*.

Omar G. Younis, Rodrigo Perez-Vicente, John U. Balis, Will Dudley, Alex Davey, and Jordan K Terry. 2024. *Minari*. `https://doi.org/10.5281/zenodo.13767625`

A Implementation Details

A.1 Custom Environment Integration

A.1.1 Wrapper Architecture

We implement a custom environment wrapper (`CustomFrankaEnv`) that integrates diffusion model predictions into the observation and reward structure. The wrapper modifies the standard Gymnasium interface to incorporate visual prediction capabilities.

Observation Space Modification: The original state-based observations are replaced with image-based observations:

- `observation`: Absolute difference between current and predicted goal images
- `achieved_goal`: Current grayscale image (120×120)
- `desired_goal`: Diffusion model predicted goal image (120×120)

A.1.2 Goal Generation Process

At each timestep, the wrapper generates goal images through the following process:

1. **Image Capture**: Render current environment state as RGB image
2. **Preprocessing**: Convert to grayscale and resize to 120×120
3. **Diffusion Inference**: Use trained diffusion model to predict future state
4. **Goal Setting**: Set predicted image as the desired goal configuration

A.1.3 Reward Function Design

The custom reward function encourages the agent to minimize the visual difference between achieved and desired states:

$$r(s_t, a_t, s_{t+1}) = -\|\text{achieved\_goal}_{t+1} - \text{desired\_goal}_t\|_2 \tag{6}$$

This formulation provides dense feedback based on visual similarity rather than sparse task completion rewards.

A.1.4 Wrapper Architecture

We implement a custom environment wrapper (`CustomFrankaEnv`) that seamlessly integrates diffusion model predictions into the observation and reward structure, fundamentally transforming the standard Gymnasium interface to incorporate visual prediction capabilities. This wrapper replaces the original state-based observations with image-based observations structured as a goal-conditioned framework, where the `observation` field contains the absolute difference between current and predicted goal images, the `achieved_goal` represents the current grayscale image at 120×120 resolution, and the `desired_goal` encompasses the diffusion model's predicted goal image at matching resolution.

A.1.5 Goal Generation Process

The goal generation mechanism operates through a coordinated sequence of operations executed at each timestep to establish meaningful visual targets for the agent. The process initiates with image capture, rendering the current environment state as an RGB image through the simulation engine's rendering pipeline. Subsequent preprocessing transforms this RGB observation through grayscale conversion and resizing to 120×120 resolution, ensuring compatibility with the trained diffusion model's expected input format. The diffusion inference stage leverages the trained model to predict the future visual state corresponding to the agent's anticipated progress, while the final goal setting phase establishes this predicted image as the desired goal configuration for the current timestep.