Extended Abstract

Motivation Imitation learning (IL) and reinforcement learning (RL) research has established algorithms that are capable of solving complex manipulation tasks, but they generally require expensive human demonstrations or hand-tuned rewards. Recent Robot Foundation Models (RFM) like Cosmos Policy have demonstrated the ability to understand the environment and zero-shot unseen tasks, but they have high compute requirements. In our work, we investigate whether Cosmos Policy can serve as a teacher to train IL and RL policies on specialized tasks, which would remove all human intervention from the loop and simplify robotics deployment.

Method We build a training pipeline with Cosmos Policy as a teacher in the loop. We first collect successful rollouts using the the teacher, which becomes demonstration data to train a student policy with Behavior Cloning (BC). We implement DAgger as a stronger IL approach to correct for distribution drifts in the student. Finally, we fine-tune the student policy using offline advantage-weighted actor-critic (AWAC) with rollouts collected from both teacher and student. For AWAC, we formulate the teacher potential reward (TPR) as a dense and high-signal reward to supplement the sparse environment reward.

Implementation We designed the student policy network to accept the same inputs as the Cosmos Policy teacher in the LIBERO simulation environment. The network consists of image and proprioception encoders feeding into an MLP body. Just like the teacher, it predicts 16-action chunks for a given state observation. DAgger is initialized on the BC checkpoint, and implements a mixed execution schedule that progressively selects student actions with higher probability as training progresses. AWAC is initialized on the DAgger checkpoint, and its offline training data is sourced from both teacher and DAgger student rollouts. Aside from actions and observations, these rollouts also include teacher annotations of value predictions. We derive the TPR from the discounted difference of value predictions before and after an action chunk. We use the same set of hyperparameters for AWAC to train on all tasks.

Results We generate 15 custom LIBERO tasks for the purpose of evaluation. For each one, we run 50 Cosmos Policy rollouts and 10 rollouts for policies trained by the BC, DAgger, and AWAC methods. We found that BC and DAgger policies does well on simple short horizon tasks, but struggles with tasks that have more steps or require higher precision. Cosmos Policy and AWAC policies both reached 98% success rates on tasks. The small student policy networks are also very cheap to run, with average inference time of under 1.58 ms, a 400 times improvement compared to 640 ms from Cosmos Policy. GPU and VRAM usage is also much lower compared to Cosmos Policy.

Discussion Our experiments show that using Cosmos Policy to teach IL and RL policies is a very effective strategy. The student policy can reach similar success rate as the teacher on task-specific benchmarks. Furthermore, the combination of the significantly lower inference time, VRAM, and GPU usage also confirms that the student model has much lower hardware consumption compared to the teacher, making it ideal for low-compute edge devices powering robots that are task specialists.

Conclusion We develop and evaluate an IL and RL pipeline that trains policies with Cosmos Policy as a teacher in the loop. Our approach addresses key issues in expert data availability and reward-tuning, making policy training fully automated. We measure that the student policies can execute 400 times faster than Cosmos Policy and use much fewer GPU resources, while reaching comparable accuracy on all simulated manipulation tasks that we tested. This shows promise as a very practical method to cheaply train and deploy policies on robots.

Training Robot Policies with a Foundation Model Teacher

Daniel Zou
Stanford Online
Stanford University
dlzou@stanford.edu

Steven Feng
Stanford Online
Stanford University
stevenf7@stanford.edu

April Yang
Stanford Online
Stanford University
aprilyyt@stanford.edu

Abstract

Robot Foundation Models demonstrate strong generalization ability on control and manipulation tasks, achieving state-of-the-art performance in many robotics benchmarks. However, their high inference cost makes them difficult to be directly deployed onto robots. Meanwhile, imitation learning (IL) and reinforcement learning (RL) techniques have been proven to work on similar tasks with smaller policy networks, but in practice, they face difficulties like human expert data acquisition or reward tuning. This paper examines the plausibility of using the robot foundation model as a teacher to automate and simplify IL/RL training of a smaller student policy. Overall, the student policies trained with our DAgger + offline RL pipeline reach a success rate of 98% across 15 LIBERO tasks, comparable to the teacher, with an inference speed of 1.58 ms per 16-action chunk, which is about 400 times faster than the teacher.

1 Introduction

The past decade of robotics research has established that imitation learning (IL) and reinforcement learning (RL) can surpass classical algorithms in solving complex control and manipulation tasks (Tang et al., 2024). However, despite many successful demonstrations of these methods in both simulated and real world tasks, they both face significant challenges in practice. IL methods require a source of expert data that is usually a team of humans, which is an expensive resource and difficult to scale. RL methods often see high instability in training—especially as the dimensionality of state and action spaces grow—and they require manually tuned rewards that are task-specific.

More recently, there has been great interest in developing robotic foundation models (RFM), which are large scale models that have been trained on extensive, diverse robotics datasets with vision, action, and language modalities. These models demonstrate emergent capability to understand physical space and actions at a semantic level, and can often solve unseen tasks in a zero-shot manner or with a small amount of fine-tuning. Cosmos Policy (Kim et al., 2026) is a recently released RFM that is based on a video diffusion architecture, and it has reported state-of-the-art success rates on generalist policy benchmarks such as LIBERO (Liu et al., 2023) and RoboCasa (Nasiriany et al., 2026). However, Cosmos Policy's 2 billion parameter network has a high inference cost that makes it difficult to deploy on real world robots with limited compute capacity.

In our work, we leverage the complementary strengths of IL/RL methods and RFMs to train lightweight policies for a variety of household tasks. Our pipeline uses Cosmos Policy as a teacher policy in two different ways. First, we use it to generate full rollouts that can be ingested as expert demonstration data. Second, we query it for value predictions on both teacher policy and student policy rollout transitions, and use them to derive the Teacher Potential Reward (TPR), which serves as a dense reward signal to augment the sparse success/failure reward from the environment. Our approach addresses the key challenges faced by both IL and RL; we can generate high volumes of

useful training data for each task that is bound by compute as opposed to humans, and then stably train student policies on a rich reward landscape without defining and tuning rewards for specific task categories. We evaluate our pipeline on new tasks generated in the LIBERO simulation framework, using a pretrained checkpoint of Cosmos Policy as the teacher. Our results show that our trained student policies can consistently approach or match the success rate of the teacher, while operating at a tiny fraction of the cost.

2 Related Work

Imitation Learning. Imitation learning (IL) applies the tried-and-true methods of supervised learning to train a model that predicts actions as high-dimensional functions of state, and it requires a source of expert data that is oftentimes a human. In the simplest setup known as behavior cloning (BC) (Bain and Sammut, 1999), an expert records a set of good demonstration trajectories, which are split into state-action pairs (s, a) in a supervised dataset. BC is simple and stable to train, but small action errors compound at test time, as the agent drifts into states that were absent from the training distribution where its predictions degrade further (Ross et al., 2011). The Dataset Aggregation (DAgger) method (Ross et al., 2011) addresses this covariate shift by improving dataset coverage of the states a policy actually visits, re-labeling them with an expert kept in the training loop. HG-DAgger (Kelly et al., 2019) reduces the human labeling burden by handing control to the expert only when the policy is uncertain or unsafe. These methods are nonetheless bounded by expensive human effort; our pipeline removes the human from the loop by using Cosmos Policy as an automated oracle.

Reinforcement Learning. Reinforcement learning (RL) optimizes a policy for long horizon rewards rather than simply mimicking an expert. In the online setting, PPO (Schulman et al., 2017) stabilizes policy-gradient updates with a clipped surrogate objective, while SAC (Haarnoja et al., 2018) is a maximum-entropy off-policy method that improves exploration and sample efficiency. In the offline setting, AWAC (Nair et al., 2020) improves a policy from a fixed dataset by weighting the policy update with the estimated advantage, constraining it toward in-distribution actions, which lets it effectively leverage large offline datasets and then fine-tune online efficiently. We adopt AWAC as the final stage of our pipeline, consuming teacher and student rollouts as offline data shaped by a dense Teacher Potential Reward.

Robot Foundation Models. Robot foundation models are large models trained on diverse robotics data that generalize to unseen tasks, either zero-shot or with lightweight fine-tuning. They primarily fall into two categories: vision-language-action (VLA) models (Physical Intelligence et al., 2025), which extend vision-language backbones to output actions, and video world model policies, which repurpose generative video models for control. The latter includes Cosmos Policy (Kim et al., 2026), which fine-tunes the Cosmos-Predict2 world model (NVIDIA et al., 2025) for visuomotor control and reports state-of-the-art performance on LIBERO (Liu et al., 2023) and RoboCasa (Nasiriany et al., 2026). Despite their strong generalization, the inference latency and compute cost of these models make them impractical as controllers deployed directly on robots, which motivates distilling their competence into a lightweight student policy.

3 Method

Our approach mainly consists of three stages: Behavior Cloning on teacher rollouts, DAgger for correcting distribution shift, and AWAC for improving success rate on more challenging tasks.

3.1 Student Network Architecture

We design a lightweight student network that operates on the same inputs as Cosmos Policy but does not use language conditioning, since each student is specialized to a single task. The architecture consists of:

- **Image encoders (×2):** A 3-layer convolutional network with Group Normalization and SiLU activations, followed by global average pooling, producing a 128-dimensional feature vector per camera view.

- **Proprioception encoder:** A single linear layer projecting the 7-dim proprioceptive state to 32 dimensions.
- **Task embedding:** A learned 32-dimensional task embedding, enabling a single checkpoint to be evaluated across tasks without language input.
- **MLP head:** A 2-layer MLP that takes the concatenated features (128 + 128 + 32 + 32 = 320 dimensions) and predicts the full 16-step action chunk ($16 \times 7 = 112$ output dimensions).

The student is roughly three orders of magnitude smaller than Cosmos Policy, making it suitable for edge deployment.

3.2 Behavior Cloning

Data Collection. For each task, we roll out Cosmos Policy for 50 episodes with random environment initializations and retain only successful trajectories. Each trajectory consists of $(o_t, a_t^{1:16})$ pairs, where o_t is the dual-camera and proprioception observation at chunk boundary t, and $a_t^{1:16}$ is the 16-step action chunk output by Cosmos Policy. This yields approximately 40–50 training episodes per task after filtering.

Training Objective. We train the student with a masked mean-squared error loss on action chunks:

$$\mathcal{L}_{\text{BC}} = \frac{1}{|\mathcal{D}|} \sum_{(o,a)\in\mathcal{D}} \sum_{k=1}^{16} \mathbf{m}_k \cdot \|\pi_\theta(o)_k - a_k\|^2, \tag{1}$$

where $\mathbf{m}_k$ is a binary mask that zeros out padding steps near episode boundaries. We optimize with AdamW (lr = 1×10^{-4}, weight decay = 1×10^{-4}) for 300 epochs, selecting the checkpoint with the lowest validation loss on a held-out 10% split.

Evaluation. After training, we evaluate the BC policy on 10 freshly initialized environments per task and report the success rate. BC serves as both a standalone baseline and the initialization for the DAgger and AWAC stages.

3.3 DAgger

Motivation. Vanilla BC trains only on teacher-visited states. At test time, the student inevitably deviates, encountering states that are out-of-distribution with respect to the training data. These small errors compound over time, leading to catastrophic failures on longer-horizon tasks. DAgger addresses this by explicitly collecting student-visited states and re-labeling them with the teacher.

Algorithm. Starting from the BC checkpoint, we run three rounds of DAgger:

1. **Mixed execution.** At each chunk boundary, we execute the teacher's action chunk with probability β and the student's action chunk with probability $1 - \beta$. We set $\beta \in \{0.5, 0.25, 0.0\}$ across the three rounds, so the student gradually takes over control.
2. **Re-labeling.** Regardless of which policy *acts*, we always query Cosmos Policy on the current observation to obtain the teacher's action chunk, and record $(o_t, a_t^{\text{teacher}})$ as a training sample. This ensures that all newly visited states receive teacher supervision.
3. **Dataset aggregation.** We collect 5 student rollouts per task per round and add the re-labeled transitions to the original BC dataset. The BC network is then retrained from scratch on the aggregated dataset using the same objective and hyperparameters.

Retraining from scratch (rather than fine-tuning from the BC checkpoint) was found to be more stable in our experiments, consistent with prior observations in the DAgger literature.

3.4 AWAC with Teacher Potential Rewards

We use offline advantage-weighted actor-critic (AWAC) as the final policy training stage after imitation learning. The offline dataset combines action trajectories from two behavior sources,

$$\mathcal{D} = \mathcal{D}_{\text{teacher}} \cup \mathcal{D}_{\text{student}}, \tag{2}$$

where teacher rollouts execute Cosmos teacher actions and student rollouts execute the DAgger-initialized student policy. In both cases, the action stored in the replay buffer is the action actually applied in the environment.

Each rollout additionally stores teacher value annotations. We write $V^*(s)$ for the Cosmos teacher lookahead potential queried from state s. This quantity is not a pure value of the current observation: Cosmos Policy generates a teacher action chunk and predicted future latent state from s, then extracts a value token from that generated sequence. Thus $V^*(s)$ is best interpreted as a teacher-conditioned potential for how promising state s is under the teacher's continuation.

We convert trajectories into chunk-level transitions of length $\ell_i \leq 16$,

$$\tau_i = \left(s_i, a_{i:i+\ell_i-1}, \widetilde{R}_i, \Gamma_i, s_{i+\ell_i}\right),$$

where $a_{i:i+\ell_i-1}$ is the executed action chunk, $R_i = \sum_{j=0}^{\ell_i-1} \gamma^j r_{i+j}$, and $\Gamma_i = 0$ for terminal chunks and $\Gamma_i = \gamma^{\ell_i}$ otherwise. When teacher values are available at both endpoints, we shape the chunk reward with a teacher potential reward (TPR), computed as the discounted difference between the two teacher values:

$$\widetilde{R}_i = R_i + \lambda_{\mathrm{TPR}} \left(\gamma^{\ell_i} V^*(s_{i+\ell_i}) - V^*(s_i)\right). \tag{3}$$

If either endpoint value is unavailable, we use the unshaped environment return R_i. For terminal chunks, no post-terminal teacher query is required: the next potential is set to 1 for success and 0 for failure or timeout. For student rollouts, $s_{i+\ell_i}$ is the actual state reached by the student action chunk, so TPR rewards student-induced progress toward states from which the teacher predicts a stronger continuation.

AWAC trains a clipped double critic on the shaped chunk return. With target critics $Q_{\bar{\theta}_k}$ and deterministic policy mean action μ_ϕ, the target is

$$y_i = \widetilde{R}_i + \Gamma_i \min_{k=1,2} Q_{\bar{\theta}_k}\left(s_{i+\ell_i}, \mu_\phi(s_{i+\ell_i})\right), \tag{4}$$

and the critic minimizes

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{\tau_i \sim \mathcal{D}}\left[\sum_{k=1}^{2} \left(Q_{\theta_k}(s_i, a_{i:i+\ell_i-1}) - y_i\right)^2\right]. \tag{5}$$

The actor is initialized from the DAgger checkpoint and trained with an advantage-weighted behavior cloning objective. We compute

$$A_i = \min_k Q_{\theta_k}(s_i, a_{i:i+\ell_i-1}) - \min_k Q_{\theta_k}(s_i, \mu_\phi(s_i)) \tag{6}$$

$$w_i = \min\left(\exp(A_i/\lambda_{\mathrm{AWAC}}), w_{\max}\right) \tag{7}$$

then optimize

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{\tau_i \sim \mathcal{D}}\left[w_i \log \pi_\phi(a_{i:i+\ell_i-1} \mid s_i)\right]. \tag{8}$$

To compute the log probability of an action $\log \pi_\phi$, the actor network has an additional head that is trained to predict the Gaussian log standard deviation of the mean action. Minibatches are sampled with replacement while balancing teacher and student rollout sources, which keeps the critic exposed to successful teacher behavior and student failure modes during offline training.

4 Experimental Setup

4.1 Training Pipeline

We conducted experiments on 15 custom generated tasks based on the LIBERO-Object and LIBERO-10 benchmark:

- **LIBERO-Object**: 1 target object with several distractor objects. The policy must correctly identify and pick up the target object and place it in the target area
- **LIBERO-10**: 1-2 target objectives. The policy must correctly do a single or multi-step task, such as turn on the stove and place the pot on the stove.

The custom tasks use the same environments and objects as official LIBERO benchmark tasks to reduce the distribution gap to how the Cosmos Policy LIBERO checkpoint was trained, but the specific task descriptions are previously unseen. Task descriptions are detailed in Appendix A.

After a task is defined, 160 initial states with different seeds are generated based on the Behavior Domain Definition Language (BDDL) specification for unique starting conditions. 50 initial states are used for the teacher roll out collection, 100 for the student roll out collection, and 10 for evaluation.

We first collect teacher rollouts using and prune out any task where the teacher has zero success rate. Then, student policies are trained using the BC and DAgger methods and evaluated using the last 10 initial states. Finally, AWAC fine-tuning is only done on tasks where the DAgger policy does not have a perfect success rate.

4.2 Evaluation

The main pipeline and benchmarks were run on an RTX PRO 6000 96GB GPU, the custom_libero_10_bowl_bottom_drawer task was run on an NVIDIA RTX A6000 48GB, so the GPU and VRAM usage are different.

In the experiments, we log the task success rate and training time between Cosmos Policy rollout, BC, DAgger, and AWAC. Additionally, we compare the per action chunk average inference time, average GPU compute usage, and average VRAM usage between each task to analyze the inference cost between the different teacher and student networks.

5 Results

Policy success rate, inference time, and GPU usage are reported in the quantitative evaluation. Policy behavior will be reported in the qualitative evaluation section.

5.1 Quantitative Evaluation

We evaluated the performance of the baseline Cosmos Policy model, as well as our BC, DAgger, and AWAC trained policies, based on three metrics: task success rate, inference time, and GPU usage during inference. These three metrics characterize a policy's task performance and compute resource usage, so we can justify the claim that our trained policies can reach a similar success rate as the baseline with a fraction of the resource usage.

5.1.1 Task Success Rate

Table 1: Success Rate across LIBERO tasks

Method	Mean Success Rate	Success Rate Standard Deviation
Cosmos policy	0.98	0.04
BC	0.76	0.27
DAgger	0.83	0.18
AWAC	0.98	0.05

The Cosmos Policy baseline performed consistently well on both simple and difficult tasks. BC performed well on simple single step tasks but had a very low success rate on the difficult LIBERO-10 tasks, hence the high standard deviation. DAgger improved upon BC success rates on most tasks, especially where BC success rate was low to begin with. With DAgger, the success rate of most simple tasks reached 100%.

AWAC further improved the success rates of the tasks to levels similar to those of the Cosmos Policy and closed the gap in both mean and standard deviation as shown in 1.

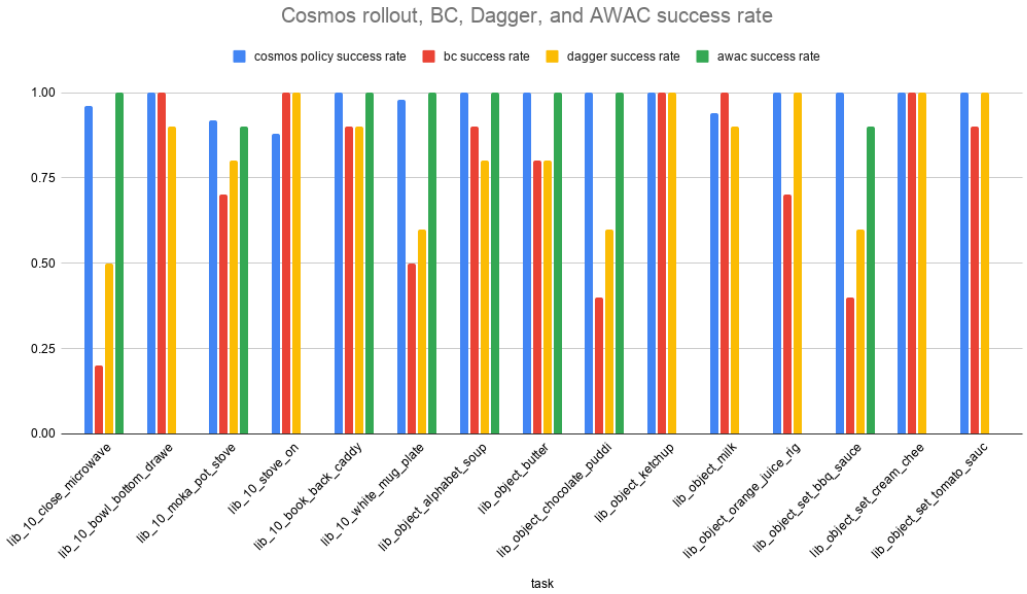

Figure 1: Cosmos Policy, BC, DAgger, and AWAC success rate

5.1.2 Inference Time

Average inference time to return an action chunk of 16 robot actions (each action contains the target position of each robot joint) was measured for Cosmos Policy, BC, DAgger, and AWAC across LIBERO tasks.

Table 2: Inference time (ms) across LIBERO tasks

Method	Mean Inference Time (ms)	Inference Time Standard Deviation (ms)
Cosmos policy	640.3	104.57
BC	1.13	0.26
DAgger	1.29	0.20
AWAC	1.58	0.06

Our smaller policy networks have a significant advantage in inference time. The time to generate an action chunk of 16 actions was around 600 milliseconds with Cosmos Policy, while the BC and DAgger trained networks took around 1.3 milliseconds. The AWAC policy takes slightly longer to execute due to an extra log-probability action head used in training, but this can be disabled for deployment. Overall, the student policies exhibit a 40,000% latency improvement compared to Cosmos Policy. See 2 for the average inference time for each task.

5.1.3 GPU Usage

We benchmarked most policies using a NVIDIA RTX PRO 6000 GPU with 96GB of VRAM to avoid performance or VRAM bottlenecks from Cosmos Policy inference. A lower GPU core usage may suggest that the model requires less GPU compute, and similarly, a lower VRAM usage indicates that the model can run on a device with less VRAM capacity, such as an edge device like the Jetson Orin.

Disclaimer: GPU usage and VRAM usage are device wide, so extra system-wide tasks such as simulation rendering may be included in the measurements. The custom_libero_10_bowl_bottom_drawer task was run on an NVIDIA RTX A6000 48GB, so the GPU and VRAM usage are different.

The student policies all use considerably less VRAM compared to Cosmos Policy at just 1GB, while Cosmos Policy requires at least 7GB as shown in 3. Furthermore, the GPU usage of student policy is

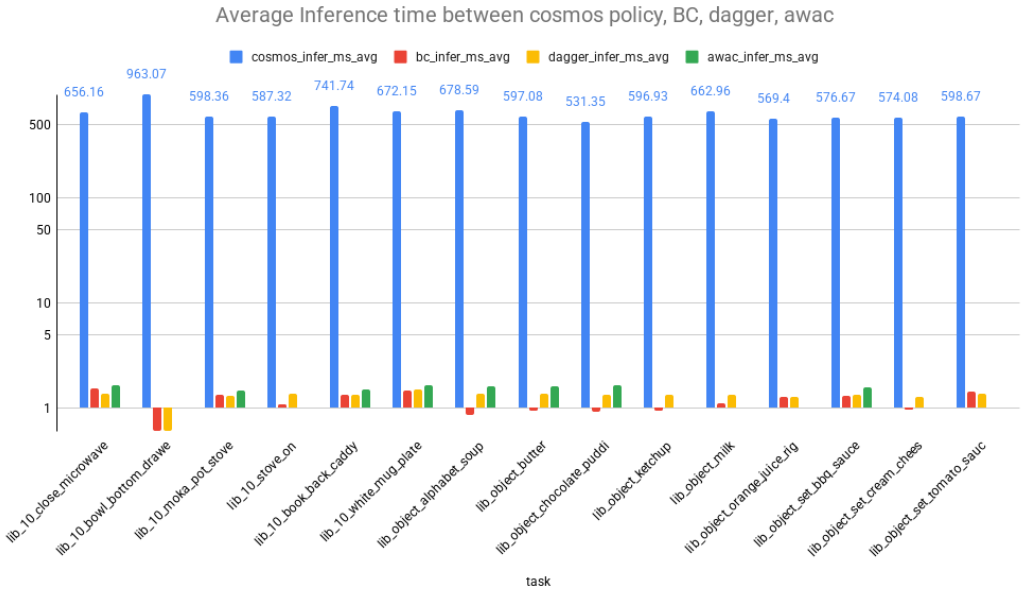

Figure 2: Cosmos Policy, BC, DAgger, and AWAC inference time

Table 3: GPU VRAM (mib) across LIBERO tasks

Method	Mean VRAM usage (mib)	VRAM usage Standard Deviation (mib)
Cosmos policy	6503.43	92.34
BC	877.70	66.68
DAgger	877.70	66.68
AWAC	925.50	5.55

also significantly lower than cosmos policy, around half of the cosmos policy usage during rollout as shown in 3

Table 4: GPU usage (%) across LIBERO tasks

Method	Mean GPU usage (%)	GPU usage Standard Deviation (%)
Cosmos policy	59.13	11.74
BC	34.08	16.37
DAgger	29.82	15.81
AWAC	24.53	6.43

5.2 Qualitative Analysis

We will use the LIBERO-10-based `close_microwave` task as an example to analyze the behavior of BC, DAgger, and AWAC. In this task, the Franka robotics arm is required to correctly pick up the target white mug from the side, place it inside an open microwave, and then close the microwave door.

5.2.1 Behavior Cloning

For a multistep task like `close_microwave`, BC is able to learn the general motions of the task sequence, such as moving the end effector toward the mug first, then moving to the microwave and closing the microwave door. However, BC alone is unable to consistently complete detailed action

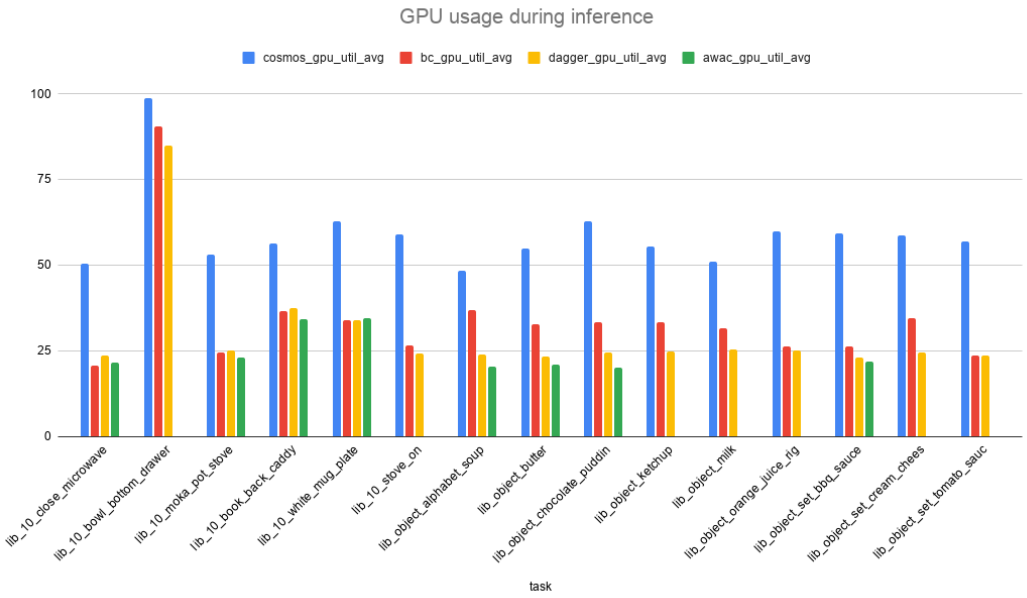

Figure 3: Cosmos Policy, BC, DAgger, and AWAC GPU Usage

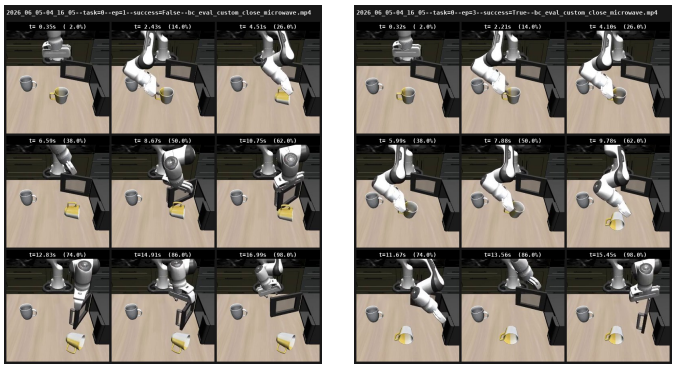

Figure 4: BC rollouts for the `close_microwave` task

checkpoints like grabbing the mug on the side to place it in the microwave. It has not been trained on states that deviate from the high quality trajectories of Cosmos Policy, so in the common event of an insecure grip being leading to the mug dropping prematurely, the arm continues the motion with no attempt at recovery and results in task failure as shown in figure 4.

5.2.2 DAgger

DAgger displays significant improvement over BC, where the arm has learned to pick up the mug from the side to move it inside the microwave, and it can adjust the end effector position if it is too high or low compared to the microwave. Another improvement is when an error is encountered, such as dropping the mug early, the policy has learned to attempt recovery, such as reorienting the mug to pick it back up. These behaviors contribute to a higher success rate at 0.5, but in many cases, the task can still end in failure as shown in figure 5.

5.2.3 AWAC

AWAC fine-tuning improves the success consistency of the policy dramatically, and the robot agent encounters fewer failure conditions. It has learned to always grab the mug precisely at the handle to

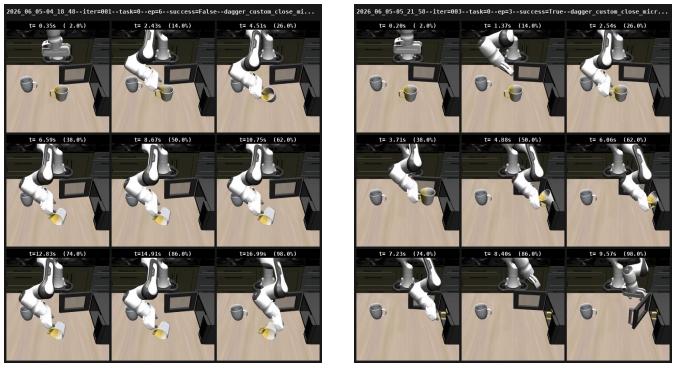

Figure 5: DAgger rollouts for the `close_microwave` task

Figure 6: AWAC rollouts for the `close_microwave` task

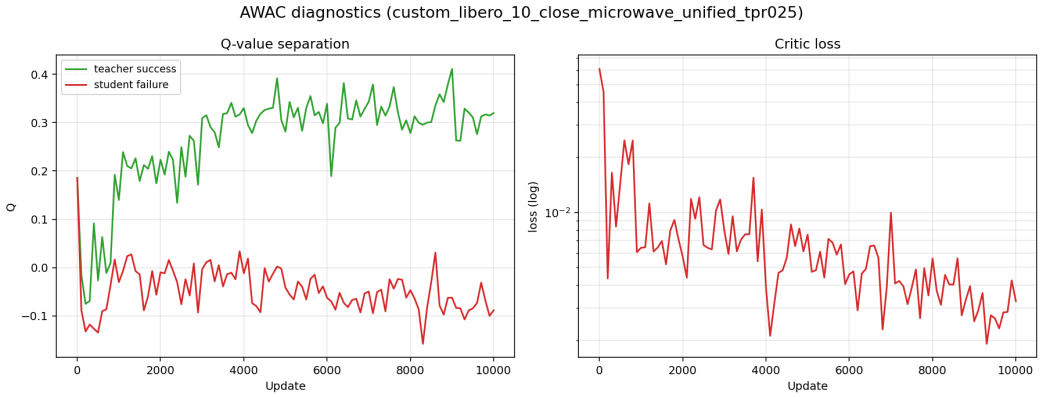

Figure 7: AWAC critic network training diagnostics

place the mug into the microwave, without hitting the interior ceiling. In our 10-trial evaluation, the AWAC policy completes the task with perfect accuracy as shown in figure 6.

Figure 7 illustrates two diagnostic measures of the critic network training process. Q-value separation shows that as training progresses, the Q-values predicted by the target critics diverge for successful and failed trajectories. Transitions that are part of successful trajectories are assigned higher Q-values on average, which means the critic has learned a good scoring metric for desirable versus undesired behaviors. Another signal that AWAC training is well-behaved is that the critic loss shows a general downward trend.

6 Discussion

6.1 Revisiting the Initial Hypothesis

We first hypothesized that we can use Cosmos Policy as a teacher to RL train a small student network on individual manipulation tasks, and at a high level, this hypothesis has been proven true. However, our exact formulation has changed significantly since the start. Our initial plan was to use Soft Actor-Critic (SAC) (Haarnoja et al., 2018), an online actor-critic method. We designed an auxiliary loss for the critic update that is intended to minimize the difference between the predicted Q-value, and a Bellman target computed from the teacher's value prediction. With this method, we were unable to train any working policy. We believe that the outputs of the teacher's value prediction head are not scaled in a way that is compatible to the SAC update objective. In addition, the SAC estimated gradient updates may contain a lot of noise due to the small volume of online data, and training could be sensitive to weight initialization and schedules of certain hyperparameters.

We found that the teacher potential reward (TPR) is critical to successful RL training. LIBERO's environment only provides a sparse reward of 1 when all goal conditions are satisfied within the step limit, or 0 otherwise. Combined with the hiigh dimensionality of the observation space in manipulation tasks, it is challenging for an actor-critic algorithm to identify good behaviors. TPR serves as a high-signal dense reward that facilitates critic training, as we previously analyzed 7.

6.2 Implications

Our experimental results show that it is not only feasible, but highly practical insert a powerful RFM teacher in the established IL/RL recipes for robot policy training. Across the 15 custom generated tasks, our pipeline consistently trained small task-specific policies that approach the success rate of the teacher model at or near 1.0, and this was achieved in a fully automated manner; no intervention was needed in tuning rewards or hyperparameters for individual tasks. Our particularly lightweight student network architecture also has negligible compute requirements compared to the Cosmos Policy teacher, which is an important advantage in edge deployment scenarios.

6.3 Limitations and Future Directions

The main limitation of our presented method is that our student policies are each specific to one task. However, in many real-world scenarios such as factory assembly lines, robots are assigned to repetitive jobs, and while they need to be robust to variations, they do not need to deeply understand the world or perform fundamentally different tasks, so the powerful generalizability of RFMs becomes redundant. Furthermore, the student training pipeline we have described is not limited to small single-task networks. One can trade more compute for a more powerful model architecture, then apply the same pipeline to easily train a more capable student policy that could, for instance, solve multiple tasks.

In this work, we focus on learning algorithms that lend themselves to offline data collection or offline RL in order to simplify Cosmos Policy inference. However, there are natural ways to extend our pipeline to online RL methods. The actor and critic networks trained during the AWAC stage can be directly reused in online actor-critic algorithms such as Soft Actor-Critic (Haarnoja et al., 2018). In the online training loop, value predictions can still be queried from the Cosmos Policy teacher to densify the reward space with TPR and simplify training. The actor and critic weights carried from AWAC give the SAC algorithm a strong initialization from which to further optimize with exploration, making it much more likely to succeed in improving the policy further. Following this procedure, it may be possible to fine-tune a student policy that exceeds the success rate of the teacher baseline.

7 Conclusion

We present a pipeline for training robot manipulation policies using IL and RL, with the novel addition of an robotic foundation model teacher—Cosmos Policy—to provide expert quality demonstration data as well as dense teacher potential rewards (TPR). In experiments on 15 custom LIBERO tasks, our completely automatable pipeline consistently trained policies that could match the success rates of the teacher, which is nearly perfect on most tasks. Prior works using similar IL/RL algorithms

require expensive human demonstrations or hand-tuned rewards. In deployment, the trained student policies have negligible compute cost compared to the RFM teacher, executing around 400 times faster on similar hardware.

The results demonstrate the advantages of our pipeline for real-world robot deployments where task space is limited to some degree, and compute hardware is constrained. This claim should be further validated by conducting sim-to-real experiments. We also outline a future direction to boost student policy performance further through online RL fine-tuning. This work lays out a practical path for large scale, low cost robot deployment using the powerful foundation models of today, and it stands to benefit as foundation models continue to improve.

8 Team Contributions

- **Daniel Zou:** Lead the development of AWAC / SAC
- **Steven Feng:** Lead the custom task environment, pipeline, and benchmark development
- **April Yang:** Lead the BC and Dagger development

Changes from Proposal We used the offline AWAC algorithm for training instead of online SAC for better results.

Acknowledgment We acknowledge the use of AI in this project (ChatGPT, Claude) for generating boilerplate code, debugging, and brainstorming. We would like to express our gratitude to NVIDIA for sponsoring GPU hardware for training and deployment.

References

Michael Bain and Claude Sammut. 1999. A Framework for Behavioural Cloning. In *Machine Intelligence 15: Intelligent Agents*, Koichi Furukawa, Donald Michie, and Stephen Muggleton (Eds.). Oxford University Press, Oxford, UK, 103–129. doi:10.1093/oso/9780198538677.003.0006

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290 [cs.LG] https://arxiv.org/abs/1801.01290

Michael Kelly, Chelsea Sidrane, Katherine Driggs-Campbell, and Mykel J. Kochenderfer. 2019. HG-DAgger: Interactive Imitation Learning with Human Experts. In *IEEE International Conference on Robotics and Automation (ICRA)*. https://arxiv.org/abs/1810.02890

Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. 2026. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. arXiv:2601.16163 [cs.AI] https://arxiv.org/abs/2601.16163

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. 2023. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. arXiv:2306.03310 [cs.AI] https://arxiv.org/abs/2306.03310

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. AWAC: Accelerating Online Reinforcement Learning with Offline Datasets. arXiv:2006.09359 [cs.LG] https://arxiv.org/abs/2006.09359

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. 2026. RoboCasa365: A Large-Scale Simulation Framework for Training and Benchmarking Generalist Robots. In *International Conference on Learning Representations (ICLR)*.

NVIDIA, Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, et al. 2025. Cosmos World Foundation Model Platform for Physical AI. arXiv:2501.03575 [cs.CV] https://arxiv.org/abs/2501.03575

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. 2025. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. arXiv:2504.16054 [cs.RO] `https://arxiv.org/abs/2504.16054`

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, Vol. 15. 627–635. `https://arxiv.org/abs/1011.0686`

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG] `https://arxiv.org/abs/1707.06347`

Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. 2024. Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes. arXiv:2408.03539 [cs.RO] `https://arxiv.org/abs/2408.03539`

A Additional Experiments

A.1 Task Descriptions

Table 5: Task Descriptions

Method	Description
custom_libero_10_close_microwave	Pick and Place the target mug in the microwave, then close microwave
custom_libero_10_bowl_bottom_drawer	Pick and Place the bowl in the bottom drawer
custom_libero_10_moka_pot_stove	Turn on the stove, then place the moka pot on the stove
custom_libero_10_stove_on	Turn on the stove
custom_libero_10_book_back_caddy	Pick and Place a brown book into the book caddy
custom_libero_10_white_mug_plate	Pick and Place the mug on the target plate
custom_libero_object_alphabet_soup	Pick and Place the alphabet soup into the basket with 5 other distractor objects.
custom_libero_object_butter	Pick and Place the butter into the basket with 5 other distractor objects.
custom_libero_object_chocolate_pudding	Pick and Place the chocolate pudding into the basket with 5 other distractor objects.
custom_libero_object_ketchup	Pick and Place the ketchup into the basket with 5 other distractor objects
custom_libero_object_milk	Pick and Place the milk into the basket with 5 other distractor objects
custom_libero_object_orange_juice_right	Pick and Place the orange juice into the basket with 5 other distractor objects
custom_libero_object_set_bbq_sauce	Pick and Place the BBQ sauce into the basket with 5 other distractor objects
custom_libero_object_set_cream_cheese	Pick and Place the cream cheese into the basket with 5 other distractor objects
custom_libero_object_set_tomato_sauce	Pick and Place the tomato sauce can into the basket with 5 other distractor objects