THE UNIVERSITY OF TENNESSEE KNOXVILLE

Leveraging Proximal Policy Optimization for Efficient Robotic Manipulation

ECE 517: Reinforcement Learning

Group 7

Shovan Chowdhury, Sk Hasibul Alam, Umarbek Guvercin, Babita Babita

Objective

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Main Goal:

- Train a robotic arm for reaching and lifting tasks using reinforcement learning (RL).

Specific Objectives:

- Develop a stable and efficient policy for robotic control.
- Use Proximal Policy Optimization (PPO) to handle high-dimensional continuous action spaces.

Background

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Current State of the Work:

- Robotics control challenges: high-dimensional action spaces, continuous control, and non-linear dynamics.
- Traditional approaches: Supervised learning and rule-based systems.
- Emergence of RL: Shift to learning policies directly from interactions with the environment.

Why Reinforcement Learning?

- Self-learning and adaptability.
- No need for explicit programming of all behaviors.

Progress in RL for Robotics:

- DDPG, SAC, and TD3 for continuous control.
- Limitations: unstable training, exploration inefficiencies.

PPO:

- Stability through clipped updates.
- Effective for complex robotic tasks.

Why PPO?

THE UNIVERSITY OF TENNESSEE KNOXVILLE

COMPARISON OF PPO WITH OTHER RL ALGORITHMS

Benchmark	DQN	A3C/A2C	TRPO	PPO
Handles continuous actions	✗	✓	✓	✓
Stability in training	✗	✗	✓	✓
Computational efficiency	✓	✓	✗	✓
Simplicity of implementation	✓	✓	✗	✓
Robustness to hyperparameters	✗	✗	✓	✓
Suitable for robotics	✗	✓	✓	✓

PPO Algorithm

THE UNIVERSITY OF TENNESSEE KNOXVILLE

During PPO learning, GAE(λ) improves the estimation of advantage function.

- Initialize advantage estimate A for all timesteps
- Calculate TD error for each timestep
- Iteratively calculate A for each timestep
- Return the normalized A

λ controls the balance between bias and variance.

Algorithm 1 Generalized Advantage Estimation (GAE)

function GAE(r, d, V, V'_{last})
  $a \leftarrow 0$
  $A \leftarrow \text{zeros}(r)$
  for reverse $i \leftarrow$ number of rows in r **do**
    if $i \neq$ last row of r **then**
      $V'_i \leftarrow V_{i+1}$
    else
      $V'_i \leftarrow V'_{last}$
    end if
    $a \leftarrow r_i - V_i + \gamma\,(\neg d_i)\,(V'_i - \lambda a)$
    $A_i \leftarrow a$
  end for
  return $R \leftarrow A + V$
  return $A \leftarrow \left(A - \bar{A}\right) / \left(A_\sigma + 10^{-8}\right)$
end function

Policy Update:

For each iteration:

- Collect, in a rollout memory, a set of states s, actions a, rewards r, dones d, log probabilities $\log p$ and values V on policy using π_θ and V_ϕ
- Estimate returns R and advantages A using GAE(λ) from the collected data $[r, d, V]$
- Compute the clipped surrogate objective (policy loss) with $ratio$ as the probability ratio between the action under the current policy and the action under the previous policy:
$$L_{\pi_\theta}^{clip} = \mathbb{E}\left[\min\left(A \cdot ratio, A \cdot \text{clip}(ratio, 1-\epsilon, 1+\epsilon)\right)\right]$$
- Compute the value loss L_{V_ϕ} as the MSE between the predicted values $V_{predicted}$ and the estimated returns R
- Optimize the total loss $L = L_{\pi_\theta}^{clip} - c_1 L_{V_\phi}$

Algorithm 2 Update for learning

$V'_{last} \leftarrow V_\phi(s')$
$R, A \leftarrow \text{GAE}(r, d, V, V'_{last})$
$[[s, a, \log p, V, R, A]] \leftarrow$ states, actions, log_prob, values, returns, advantages ▷ sample mini-batches
for all learning epochs **do**
  for all mini batches **do**
    $\log p' \leftarrow \pi_\theta(s, a)$
    $ratio \leftarrow \log p' - \log p$
    $KL_{divergence} \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left(e^{ratio} - 1 - ratio\right)$
    if $KL_{divergence} > KL_{threshold}$ **then**
      break
    end if
    $ratio \leftarrow e^{\log p' - \log p}$
    $L_{surrogate} \leftarrow A\ ratio$
    $L_{clipped} \leftarrow A\ \text{clip}(ratio, 1-\epsilon, 1+\epsilon)$
    $L_{\pi_\theta}^{clip} \leftarrow -\frac{1}{N}\sum_{i=1}^{N}\min\left(L_{surrogate}, L_{clipped}\right)$
    $V_{predicted} \leftarrow V_\phi(s)$
    $V_{predicted} \leftarrow V + \text{clip}\left(V_{predicted} - V, -\beta, \beta\right)$
    $L_{V_\phi} \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left(R - V_{predicted}\right)^2$
    reset $\text{OPTIMIZER}_{\theta,\phi}$ ▷ torch Adam
    $\nabla_{\theta,\phi}\left(L_{\pi_\theta}^{clip} + L_{V_\phi}\right)$
    clip_grad_norm$(\|\nabla_{\theta,\phi}\|)$
    step $\text{OPTIMIZER}_{\theta,\phi}$
  end for
end for

Methodology

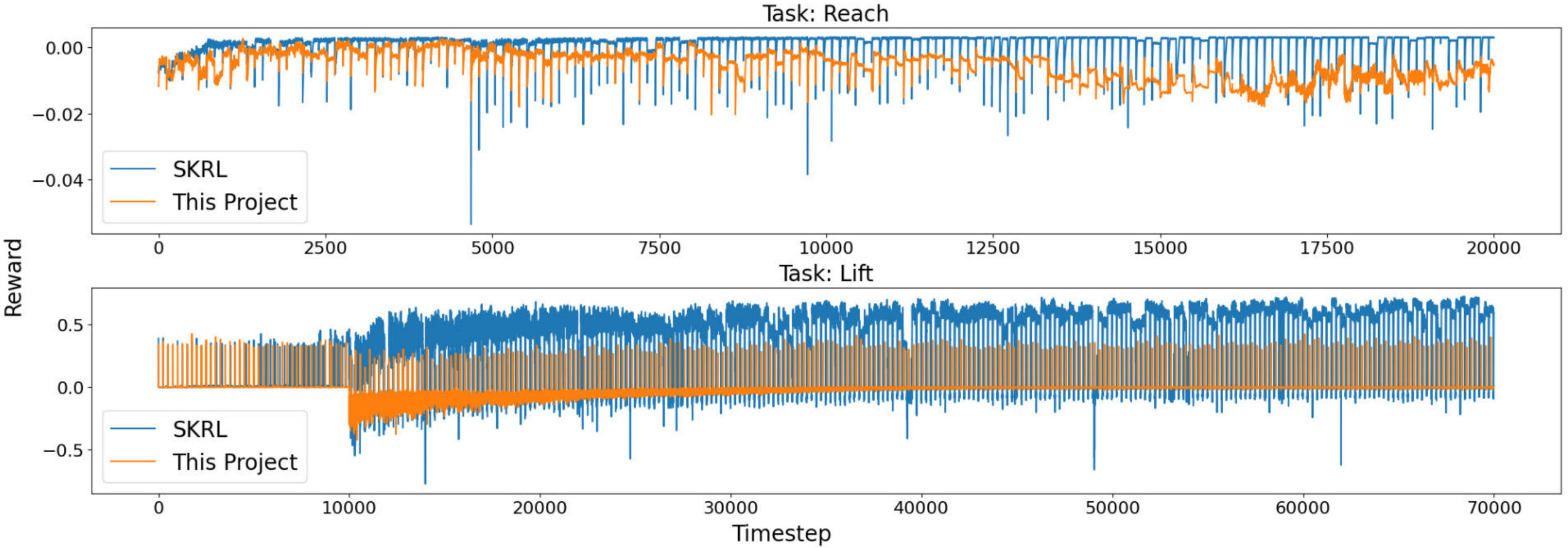
- Environments
  - `Isaac-Reach-Franka-v0`
  - `Isaac-Lift-Cube-Franka-v0`
- Memories
  - Rollout buffers → Random memories from SKRL
- Models
  - Agent's brain → Neural networks as function approximator
- Reward

Reach		Lift	
Name	**Weight**	**Name**	**Weight**
end_effector_pos_track	−0.2	reaching_object	+1
end_effector_pos_track_fine	+0.1	lifting_object	+15
end_effector_orientation_track	−0.1	obj_goal_track	+16
action_rate	−0.0001	obj_goal_track_fine	+5
joint_vel	−0.0001	action_rate	−0.0001
		joint_vel	−0.0001

- Training [*50 parallel instances*]
  - 20000 timesteps for `Reach`
  - 70000 timesteps for `Lift`
- Evaluation [*1 instance*]
  - 2000 timesteps → Video rendered

Results

(1) Learning Rate Scheduler *and* Preprocessor

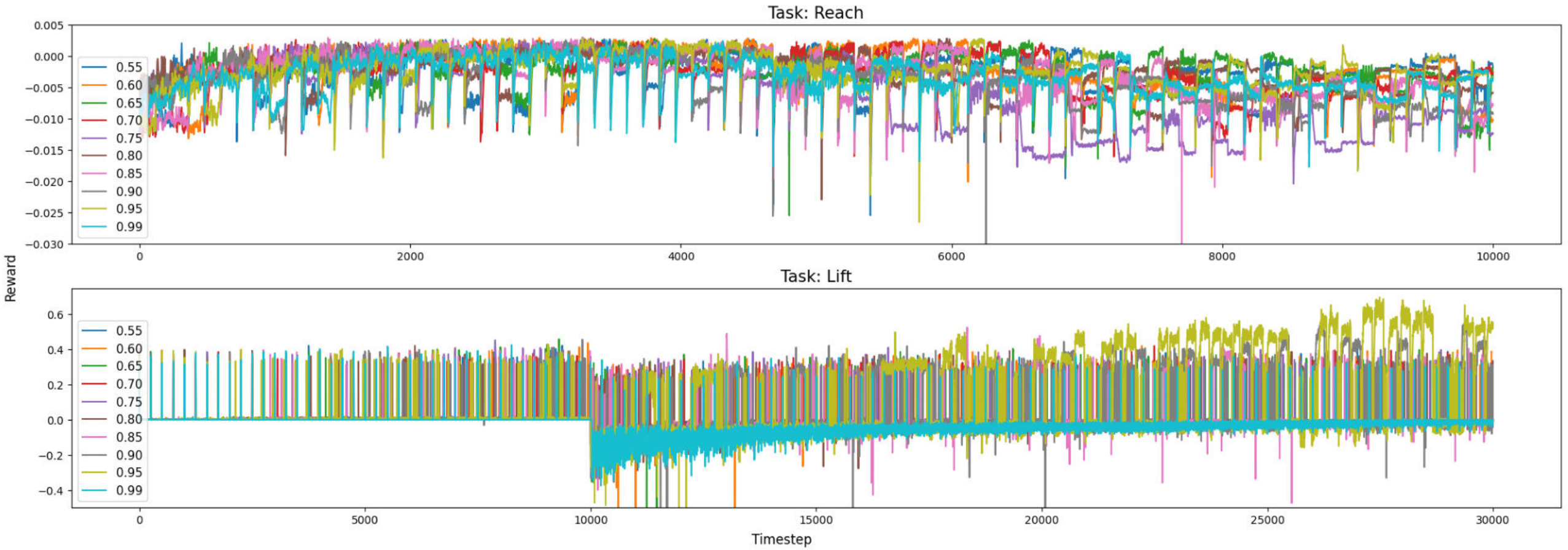


Takeaway:

- High η helps earlier in training (exploration). Low η helps later in training (exploitation).
- NN used in PPO may work better when inputs are standardized.

Results

(2) Discount Factor

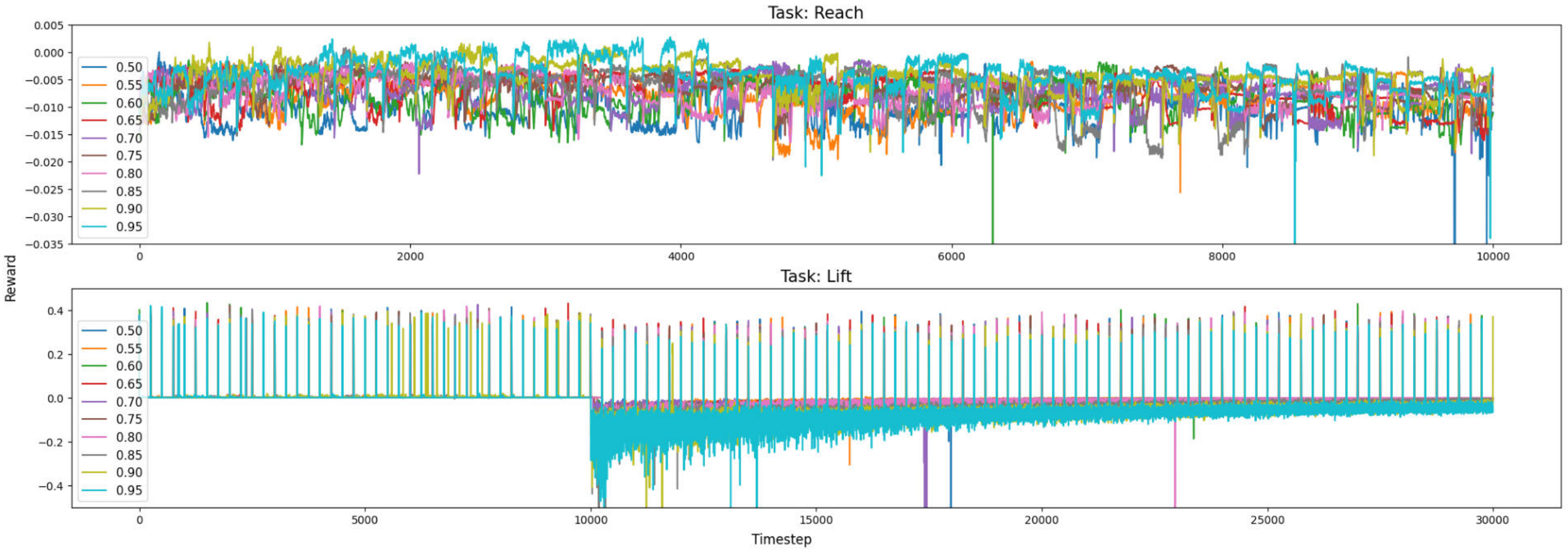


Takeaway:

- High γ considers long-term rewards (exploration), excels in sparse reward env. [→ helped us]
- Low γ considers immediate rewards (exploitation), struggles in sparse reward env.

Results

(3) TD Coefficient

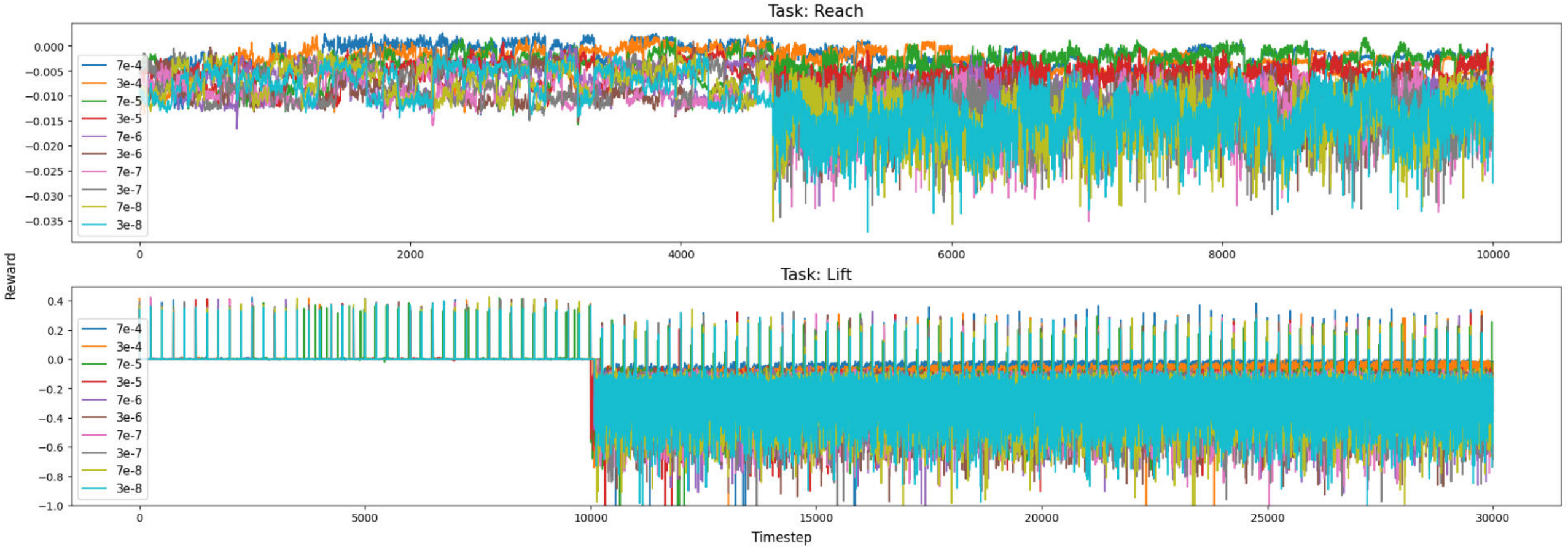


Takeaway:

- High λ considers long-term rewards (exploration), reduces bias. [→ helped us]
- Low λ considers immediate rewards (exploitation), reduces variance.

Results

(4) Learning Rate

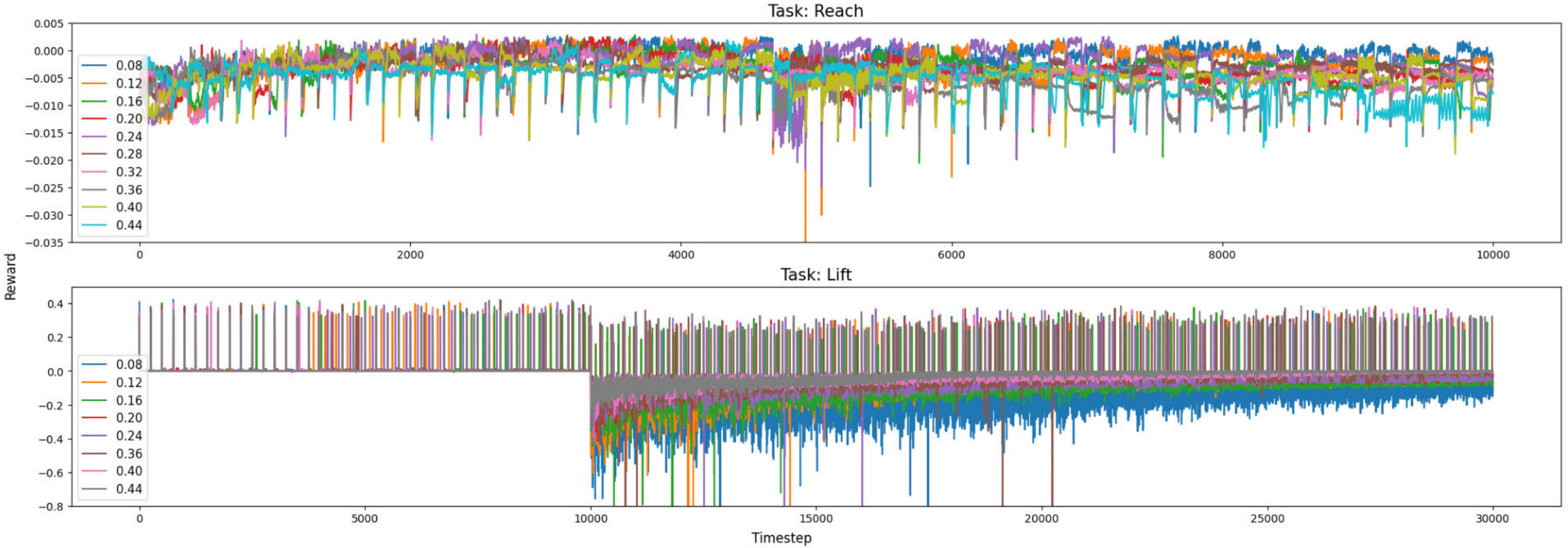


Takeaway:

- High η promotes rapid policy change (exploration), converges faster. [→ helped us]
- Low η promotes gradual policy refinement (exploitation), requires more timesteps.

Results

(5) Clip Ratio



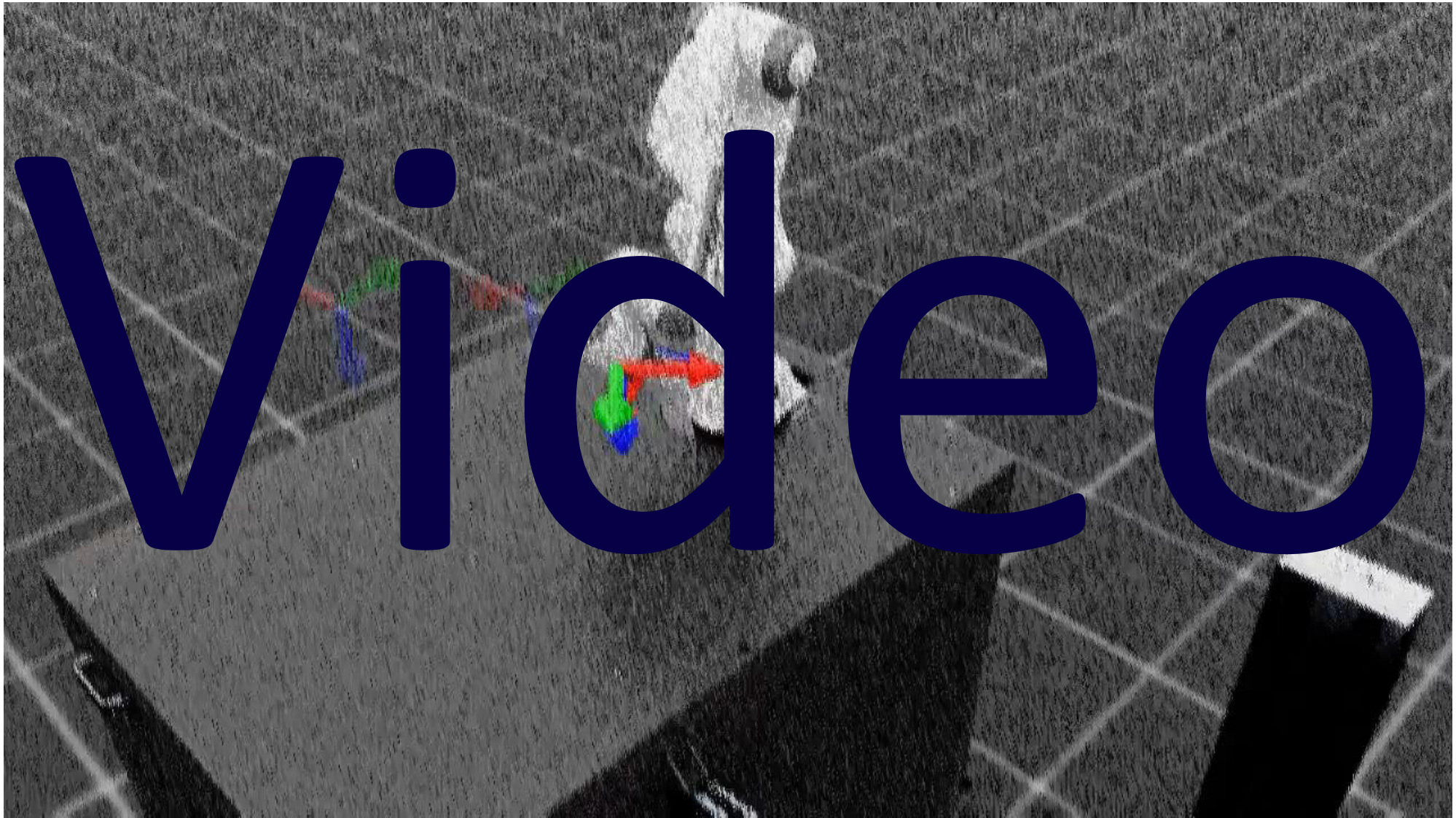
Takeaway:

- High ϵ allows larger policy update (exploration), risks overshooting.
- Low ϵ prevents large change (exploitation), mitigates noisy estimates. [→ helped us]

Evaluation Video

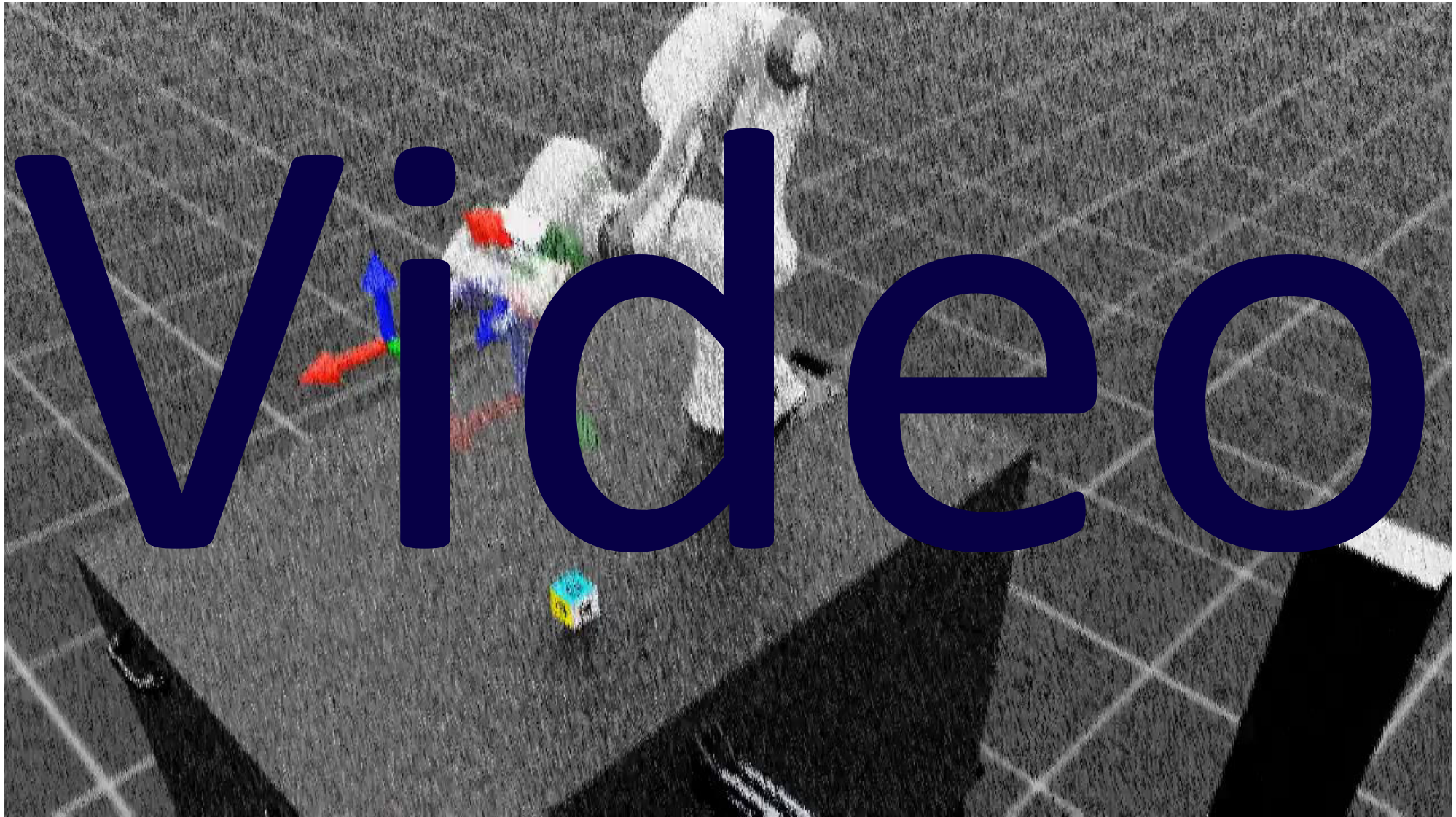
THE UNIVERSITY OF TENNESSEE KNOXVILLE

(1) Reach



Evaluation Video

(2) `Lift`



Conclusion

- The project was a success!
  - η-scheduler and preprocessor might have helped.
- Goal: Explore various hyperparameters
  - not reinvent the PPO algorithm itself
- Higher values of γ, λ, η helped.
  - promoted exploration, but increased variance
- Lower value of ϵ helped.
  - promoted exploitation, but required more iterations

Contribution

- Shovan – memory and models, literature review, writing report and preparing slides
- Hasibul – interfacing modules, exploring hyperparameters, rendering video, writing report, preparing slides
- Umarbek – PPO update function, report
- Babita – GAE and PPO update function, report

Thank you!

Questions? Comments? Concerns?