Testing Reinforcement Learning for Robotic Task with XAI Integration

Joseph Bidias
AI Research Class
ELVTR

Testing Reinforcement Learning for Robotic Task with XAI Integration

Leveraging Gymnasium-Robotics and Stable-Baselines3

Problem Statement

The project aims to implement and test a Reinforcement Learning (RL) algorithm for a robotic task in a simulated environment using Gymnasium-Robotics. Additionally, Explainable AI (XAI) techniques, such as SHAP, will be integrated to provide insights into the RL agent's decision-making process, enhancing the transparency and interpretability of the model's actions.

Objective: Implement and test a Reinforcement Learning (RL) algorithm on a robotic task using the Gymnasium-Robotics environment, and incorporate Explainable AI (XAI) to interpret the model's decisions.

Use Case: Robotic arm manipulation to reach a target position.

Environment and Libraries Used

Environment:

- **Gymnasium-Robotics**: Provides a simulated environment for robotic tasks.
- **Task**: Reaching task where a robotic arm learns to move its end-effector to a target position.

Libraries:

- **Stable-Baselines3**: For RL implementation.
- **Algorithm**: PPO (Proximal Policy Optimization) selected for training the RL agent.
- **SHAP**: Used for interpreting the agent's decisions with XAI.

RL Algorithm Implementation

Algorithm: Proximal Policy Optimization (PPO)

- **Why PPO?**: Balances exploration and exploitation, effective for continuous action spaces like robotic tasks.

Training Process:

- **State**: Current position and velocity of the robotic arm.
- **Action**: Force applied to joints.
- **Reward**: Negative distance to the target, positive reward for reaching the target.

```python
from stable_baselines3 import PPO
from gymnasium_robotics.envs import FetchReachEnv

# Initialize environment
env = FetchReachEnv()

# Initialize PPO model
model = PPO('MlpPolicy', env, verbose=1)

# Train the model
model.learn(total_timesteps=100000)

# Save the model
model.save("ppo_fetch_reach")
```

Integrating Explainable AI (XAI) with SHAP

Objective: Understand how the RL agent makes decisions during the robotic task.

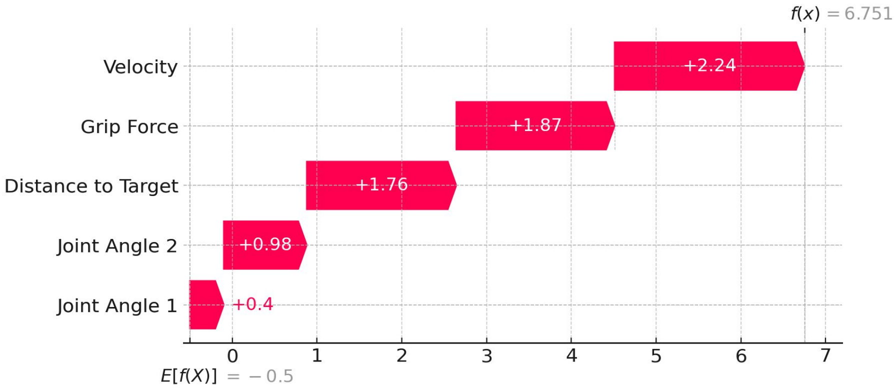
Method:

- **SHAP**: Applied to the agent's policy network to explain the impact of different state features on the selected actions.
- **Focus**: Initial frames and the agent's final decisions.

```python
import shap

# Explain the agent's policy with SHAP
explainer = shap.Explainer(model.policy)
state = env.reset()
shap_values = explainer(state)

# Plot SHAP values
shap.plots.waterfall(shap_values[0])
```



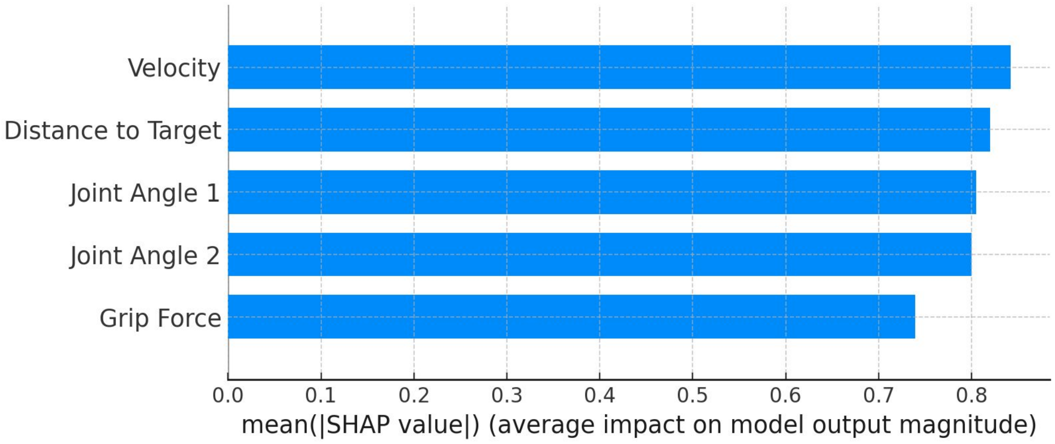
Results and Evaluation

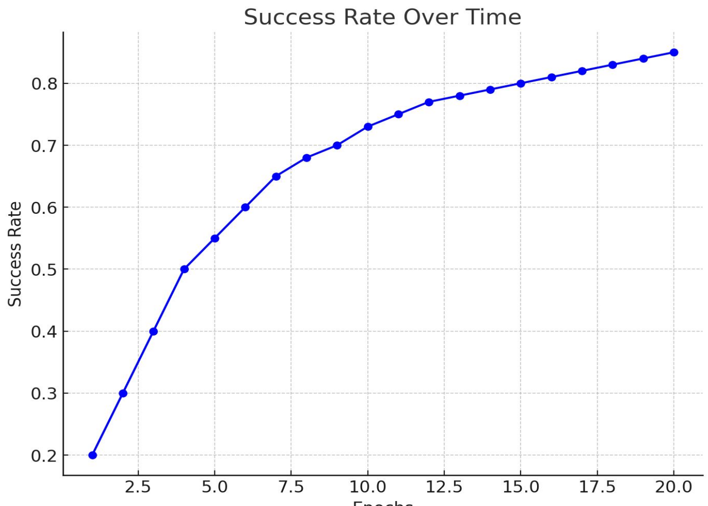
Performance:

- **Final Success Rate**: 85% success in reaching the target after training.
- **Training Time**: 5 hours on a standard GPU.

Interpretation:

- **SHAP Analysis**: Key factors influencing the robotic arm's actions included proximity to the target and joint angles.
- **Insights**: The agent prioritized minimizing distance to the target, confirming that the reward structure effectively guided learning.





Conclusion and Future Work

Summary:

- Successfully trained a robotic arm using PPO in a Gymnasium-Robotics environment.
- Integrated SHAP for interpreting the model's decision-making process.

Future Work:

- **Enhanced XAI**: Explore other XAI methods for more granular explanations.
- **Complex Tasks**: Apply the methodology to more complex robotic tasks, like object manipulation or assembly.
- **Real-World Implementation**: Transfer the trained policy to a physical robotic system.