Policy Gradient for CartPole

Introduction:

This report provides an in-depth analysis of training an agent to solve the CartPole problem using the REINFORCE algorithm, a policy gradient method. The CartPole problem is a fundamental benchmark in reinforcement learning where the agent's goal is to balance a pole on a cart by taking appropriate actions to prevent it from falling.

Training Implementation Details:

- **Algorithm:** The REINFORCE algorithm is employed to optimize the agent's policy through policy gradient updates. The agent interacts with the environment and collects episode data to compute cumulative rewards and policy loss for updating the policy network.

Policy Network Architecture:

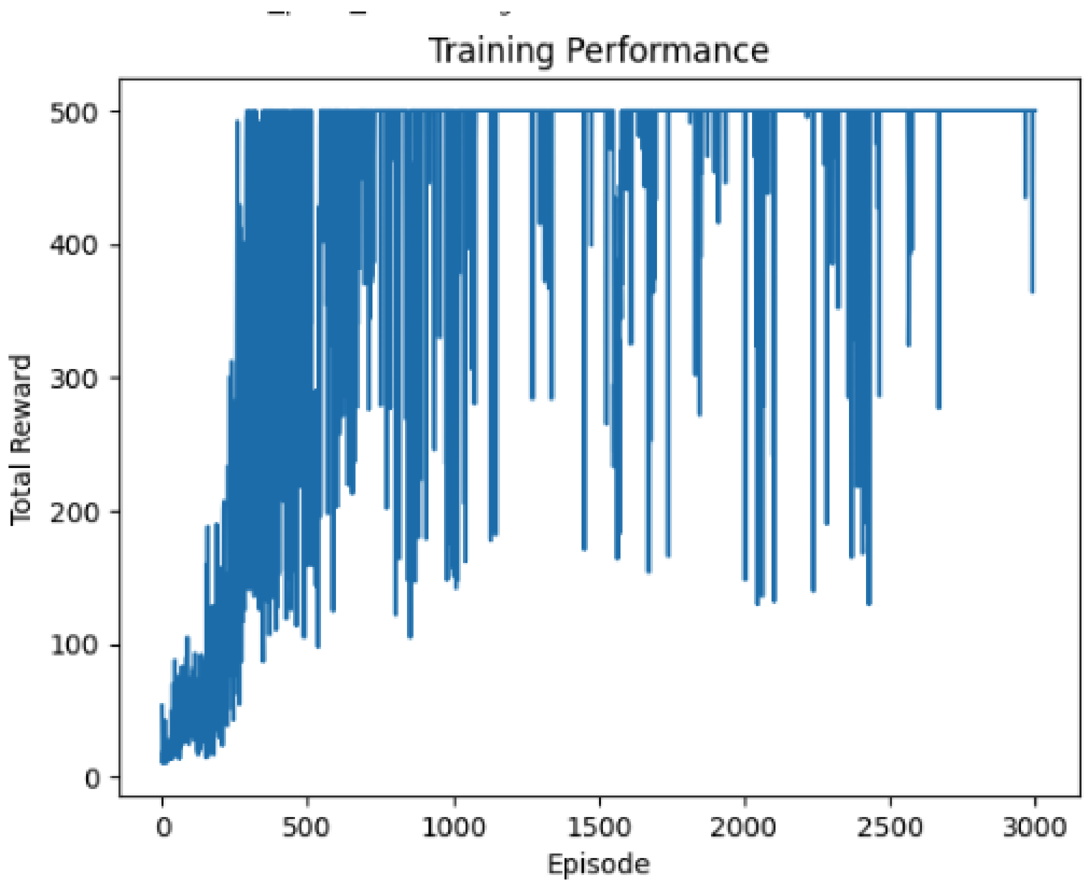
- **Input Layer:** Takes a 4-dimensional state vector representing cart position, cart velocity, pole angle, and pole tip velocity.
- **Hidden Layer:** Contains 128 neurons with ReLU activation to introduce non-linearity and learn complex patterns.
- **Output Layer:** Utilizes Softmax activation to output action probabilities for deciding between moving left or right.

Training Setup:

- **Total Episodes:** 3000 episodes were used for training the agent.
- **Optimizer:** The policy network was trained using the Adam optimizer with a learning rate of 0.001.
- **Reward Normalization:** Applied to reduce gradient variance and enhance training stability.
- **Discount Factor (gamma):** Set to 0.99 to emphasize future rewards while balancing immediate returns.

Plot Analysis:

The attached training plot illustrates the total rewards obtained per episode over the entire training duration.



1. **Initial Phase (0 to ~500 episodes):**
   - The early phase shows a rapid increase in total rewards as the agent begins to learn effective strategies for balancing the pole. The steep slope in this part of the curve indicates successful initial learning and policy improvements.
   - Rewards start at low values (e.g., below 100) and gradually increase, showing that the agent is becoming more adept at prolonging the pole's balance.
2. **Intermediate Phase (~500 to ~1000 episodes):**
   - The agent reaches episodes with rewards close to the maximum of 500. This suggests that the policy can balance the pole for long durations.
   - However, some variability is noted with occasional dips in performance, indicating moments when the policy may encounter states leading to suboptimal actions or failure to maintain balance.
3. **Stabilization and High Variability Phase (~1000 to 3000 episodes):**
   - In this phase, the plot shows frequent instances where the agent reaches the maximum reward (500), demonstrating that the policy is effective in many episodes.
   - Despite high performance, there are visible fluctuations where total rewards decrease significantly in certain episodes. These drops could result from exploration strategies, the stochastic nature of policy decisions, or suboptimal state transitions.

- The overall trend still showcases a well-learned policy, but the variability suggests room for further fine-tuning to achieve more consistent results.

Observations and Insights:

- **Maximum Achieved Reward:** The agent successfully attains the maximum possible reward (500) in numerous episodes, indicating effective learning and a well-optimized policy.
- **Minimum Observed Reward:** Occasional episodes with significantly lower rewards point to policy inconsistency, potentially due to exploration or the random nature of action sampling.
- **Stability:** The variability in rewards during later episodes suggests that while the policy performs well overall, incorporating techniques to reduce variance could enhance consistency.

Recommendations for Improvement:

1. **Hyperparameter Tuning:**
    - Reducing the learning rate may help to make policy updates smoother and reduce variability in training.
    - Adjusting the discount factor (gamma) can influence the agent's prioritization of long-term versus immediate rewards, potentially stabilizing training.
2. **Baseline Techniques:**
    - Introducing more sophisticated baselines, such as a learned value function, could reduce the variance of policy gradient estimates and improve stability.
3. **Exploration and Regularization:**
    - Incorporating entropy regularization could encourage the agent to explore more during training and avoid overly deterministic policies.
    - Adaptive noise strategies can help balance exploration and exploitation, leading to more consistent performance.
4. **Extended Training and Early Stopping:**
    - Training for additional episodes with an early stopping mechanism based on reward trends could allow the agent to converge more effectively while preventing overfitting.

Conclusion:

The CartPole project demonstrated a successful implementation of the REINFORCE algorithm using PyTorch. The agent effectively learned to balance the pole and achieved high rewards in many episodes, as shown in the training plot. The occasional performance drops highlight areas for improvement, but the overall results confirm that the policy gradient method is suitable for solving this type of reinforcement learning problem. Future work can focus on enhancing stability and reducing variability for even better performance.