COMS7071A Assignment: Grid2Op using DQN and PPO

Willem Van Der Merwe 2914429

University of the Witwatersrand, Johannesburg

21 October 2024

Chapter 1

Model Architectures

The reinforcement learning algorithms selected for this study are Proximal Policy Optimization (PPO) and Deep Q-Network (DQN). These algorithms will be implemented and iteratively improved to optimize power grid operations in the Grid2Op environment.

1.1 Deep Q-Network (DQN)

DQN is a value-based method that approximates the optimal action-value function using deep neural networks, which is suitable for handling high-dimensional state spaces. DQN relies on a Q-learning framework combined with a deep neural network to approximate the optimal Q-values for each state-action pair, which enables the agent to make decisions that maximize expected rewards. Due to this algorithm already being part of prior assessments I don't cover the architecture indetail.

1.2 Proximal Policy Optimization (PPO)

PPO is a policy gradient method designed to optimize a parameterized policy by balancing exploration and exploitation while avoiding large, destabilizing policy updates. PPO's stability and efficiency stem from its use of a clipped surrogate objective function to limit the extent of each policy update, ensuring that learning is reliable without excessive policy divergence.

PPO belongs to the category of policy optimization methods and utilizes an Actor-Critic architecture, which involves updating both the policy (actor) and the value function (critic). The primary goal of PPO is to constrain policy updates to

prevent large steps that can destabilize learning. Unlike traditional policy gradient methods, PPO uses a clipped surrogate objective to regulate policy updates, thus ensuring stable and reliable training. Below, the structure and workings of PPO are detailed:

Actor-Critic Architecture : PPO uses an actor network, which outputs a probability distribution over actions, and a critic network, which estimates the value function—the expected return from a given state. The actor explores the action space by selecting actions probabilistically, while the critic provides feedback on the quality of the selected actions, thereby guiding the actor's learning.

Clipped Surrogate Objective : PPO maintains stability through the use of a clipped surrogate objective function. The objective function for PPO is given by:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

Here, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the ratio of the probability of selecting action a_t under the updated policy to the probability under the old policy, and $\hat{A}_t$ represents the advantage function. The advantage function quantifies how much better the chosen action is compared to the average action for the given state. The clipping operation constrains $r_t(\theta)$ to lie within the interval $[1-\epsilon, 1+\epsilon]$, preventing updates that are excessively large or too small, thus keeping learning within a stable range.

Advantage Estimation : PPO employs Generalized Advantage Estimation (GAE) to compute the advantage function $\hat{A}_t$. GAE is useful for reducing the variance in policy gradient estimates while preserving their accuracy, leading to more stable and efficient learning.

$$\hat{A}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

Where γ is the discount factor, and λ controls the bias-variance tradeoff. The temporal difference (TD) error δ_t measures how well the value function estimates the return compared to the observed outcome.

Entropy Regularization : Entropy regularization is employed in PPO to ensure that the policy maintains adequate exploration throughout the learning process. By adding an entropy term to the objective function, PPO encourages the policy to remain diverse in its action selection, thus preventing premature convergence to suboptimal strategies.

Policy Updates : The policy is updated using stochastic gradient ascent, aiming to maximize the clipped surrogate objective. To ensure stable and gradual learning, PPO performs multiple epochs of minibatch updates with a small learning rate. This process ensures that the policy does not change too drastically in any one update.

Implementation Details : During training, PPO alternates between collecting data by running the current policy in the environment and optimizing the clipped surrogate objective using the collected data. Key hyperparameters include the clipping threshold ϵ, the discount factor γ, the GAE parameter λ, and the number of epochs per iteration. Proper tuning of these hyperparameters is critical to the success of PPO in achieving stable learning.

Chapter 2

Model Improvements architectures

The methodology involves an iterative improvement process for both PPO and DQN. Initially, a base version of each algorithm will be implemented. Subsequently, three iterations of improvements will be applied to both agents. Each improvement is designed to incrementally enhance agent performance and robustness, allowing for an in-depth analysis of the impact of each enhancement.

2.1 DQN Improvements

Iteration 1 Implement Double DQN to mitigate overestimation bias in action-value estimation, leading to more accurate Q-value predictions and enhanced policy quality.

Double DQN addresses a fundamental limitation of the original DQN: overestimation bias in action-value predictions. In traditional DQN, the same network is used for both action selection and evaluation, which tends to result in overly optimistic value estimates. Double DQN resolves this by using two separate networks—one to select actions and the other to evaluate their value.

The key difference lies in how the target values are calculated. Instead of using the maximum Q-value estimated by the same network, Double DQN uses the main network to select the action and the target network to evaluate its value. This decoupling of action selection from action evaluation reduces the upward bias and results in more accurate value estimates.

The implementation of Double DQN involves maintaining two neural networks: the primary Q-network and the target Q-network. The target network is updated periodically, providing a stable reference point for value calculations, while the

main network is updated at each step. This helps maintain stability in the learning process by preventing rapid shifts in target estimates.

The impact of Double DQN on training output is significant, as it reduces the likelihood of divergence and leads to more reliable policy learning. The reduced bias in value estimation contributes to smoother convergence and a more stable policy, which is particularly advantageous in complex environments like power grid management.

Iteration 2 : Implement Dueling DQN to differentiate between the value of being in a state and the advantage of selecting specific actions, thereby enhancing learning efficiency.

Dueling DQN introduces a novel architecture that separates the estimation of the state-value function from the advantage function. The state-value function estimates the value of being in a particular state, while the advantage function estimates the relative benefit of each possible action compared to an average action in that state.

The key concept is that not all actions are equally important in every state. In many situations, the value of a state is independent of the specific action taken. By having separate streams for state-value and advantage estimation, Dueling DQN allows the network to focus on learning the value of the state itself without being overly influenced by the choice of action.

Implementation involves modifying the network architecture to create two separate output streams: one for the state-value and one for the advantage function. These streams are then combined to compute the Q-values for each action. This architectural change allows the network to better differentiate between states that are generally good and actions that provide additional benefits, which accelerates the learning process.

The output of Dueling DQN is a more efficient learning process with faster convergence. The separation of value and advantage leads to improved stability, particularly in environments where many actions yield similar outcomes. This allows the agent to better generalize across states and actions, resulting in more robust policies.

Iteration 3 : Integrate Prioritized Experience Replay (PER) to improve learning efficiency by sampling important transitions more frequently, thereby focusing updates on the most informative experiences.

Prioritized Experience Replay (PER) enhances the efficiency of experience replay by assigning a priority value to each transition in the replay buffer, based on its temporal difference (TD) error. Transitions with larger TD errors indicate greater learning potential, as they represent situations where the agent's prediction was significantly incorrect.

The core idea behind PER is to focus the learning process on the experiences that will have the greatest impact on improving the policy. By prioritizing transitions with high TD errors, the agent can more effectively correct its mistakes, leading to faster learning and improved policy quality.

Implementation of PER requires modifying the replay buffer to maintain priority values for each transition. During sampling, transitions with higher priority are more likely to be replayed, ensuring that the agent focuses on learning from its most informative experiences. Additionally, importance sampling weights are used to correct any bias introduced by the prioritized sampling process, ensuring stable updates.

The impact of PER on output is significant, as it accelerates the convergence of the learning process. By focusing on key experiences, the agent is able to learn more effectively from limited data, which is particularly advantageous in environments with sparse rewards. This leads to more effective policy updates, reduced training time, and better overall performance.

2.2 PPO Improvements

Iteration 1 :

Implement the "Truly PPO" methodology as described in the 2020 paper by Yuhui Wang, Hao He, and Xiaoyang Tan. This iteration involves introducing a more adaptive approach to clipping, refined hyperparameter tuning, and improved balance between exploration and exploitation.

The "Truly PPO" version aims to address limitations of the original PPO by enhancing the adaptability of the clipping mechanism. The clipping threshold is dynamic rather than fixed, enabling policy updates that are more context-sensitive.

This adaptability results in more reliable learning, particularly in environments with diverse state distributions.

The core enhancement lies in how the surrogate objective is managed. By making the clipping region adaptive, the agent can maintain a balance between stability and the capacity for significant learning updates. This modification allows the policy to evolve naturally, without the risk of becoming overly constrained or excessively volatile.

Implementing "Truly PPO" necessitates careful hyperparameter adjustment to ensure optimal clipping behavior, which introduces computational overhead but yields significant improvements in policy stability and convergence. Extensive evaluation is needed to understand the impact of these adjustments across various phases of the training cycle.

Overall, "Truly PPO" provides a framework that facilitates smoother, more stable updates, contributing to enhanced reward maximization over the long term.

Iteration 2 :

Incorporate the Intrinsic Curiosity Module (ICM) to enhance exploration by introducing a curiosity-based reward signal. ICM models the prediction error of transitioning to new states, thereby fostering exploration in less-visited areas of the state space.

The Intrinsic Curiosity Module augments the standard reward signal with an intrinsic reward based on prediction error. The ICM consists of a forward model and an inverse model. The forward model predicts the next state given the current state and action, while the inverse model predicts the action given consecutive states. The magnitude of the prediction error from the forward model serves as the intrinsic reward, which encourages the agent to visit states where it has a high level of uncertainty.

The primary goal of ICM is to motivate exploration, particularly in environments where external rewards are sparse or delayed. This added incentive helps the agent discover new strategies that could yield better results. In power grid operations, ICM might encourage the agent to explore a wider range of control actions, which could lead to more robust grid management strategies.

From an implementation perspective, integrating ICM requires modifying the neural network architecture to include the forward and inverse models, increasing

computational complexity and training requirements. However, this added complexity is justified by the resulting improvements in exploration efficiency, leading to more robust policy performance.

By encouraging exploration through intrinsic motivation, ICM helps mitigate the problem of premature convergence and ensures a more exhaustive search of the state-action space, ultimately resulting in a better-performing policy.

Note : No improvement 3 for ppo. A Candidate improvement was the novel ..... self imitation learning adaptation, the source code is included in the submission however results and runtimes not covered in the report.

Chapter 3

Runtime And Configuration

3.1 Environment Setup

The Grid2Op environment will be used to simulate power grid operations. Specifically, the "l2rpn_case14_sandbox" environment is chosen for its simplicity and focus on power flow management without adversarial components. The observation and action spaces will be adapted to suit both PPO and DQN agents, ensuring that the agents can effectively interact with the environment. The reward function used in this study is a **CombinedScaledReward**, which is a weighted combination of the L2RPN Reward and the N-1 Reward. This combined reward encourages the agents to maximize power flow while maintaining grid resilience by ensuring there are no single points of failure.

3.2 Train Time

3.3 Inference

3.4 Hyperparameters

Chapter 4

Evaluations

Each iteration will follow the same structure for evaluation:

- **Iteration 1**: Evaluate the base agent's performance and analyze the impact of the first improvement. - **Iteration 2**: Assess the agent after the next improvement and compare against previous results. - **Iteration 3**: Final evaluation after the last improvement, summarizing overall progress and effectiveness.