Reinforcement Learning Assignment

Sani Abdullahi Sani: 2770930
Christine Mtaranyika: 2770653
Liam Culligan: 0715293H
`https://github.com/liamculligan/wits_rl_project`

1 Introduction

Electricity is fundamental to modern life, powering homes, industries, and infrastructure. It plays a crucial role in driving economic growth and development [1], [2]. Power grids, which are complex systems responsible for transporting electricity from generation sources to consumers over long distances, are central to this process. These grids rely on high-voltage power lines, substations, and transformers to manage the flow of electricity and ensure that supply meets demand at all times. Ensuring the reliability of power grids is a significant challenge because disruptions or failures can result in widespread blackouts, impacting millions of people and causing major economic losses [3].

The successful operation of a power grid requires adhering to several critical constraints [3]:

- Keeping electricity flowing within the thermal limits of transmission equipment.
- Maintaining voltage levels across different regions of the grid.
- Ensuring the load on the grid matches the power generated at all times.

The complexity of managing such grids has led to increased research in advanced techniques, like reinforcement learning (RL), to help operators make faster and more effective decisions [4], [5]. The Learning to Run a Power Network (L2RPN) challenge was designed to explore the use of RL agents to manage real-time power grid operations. These environments simulate sequential decision-making problems where an RL agent must control the power network's structure and balance power flows to avoid overloads and blackouts [4], [5]. The various environments used for LR2PN are built using the Grid2Op framework [6], which allows agents to interact with a power network through a standard OpenAI Gym interface.

1.1 Environment

The `l2rpn_case14_sandbox` [6] is a simplified environment used in the L2RPN challenge, designed to focus on managing power grid flows without external disruptions such as adversarial opponents or maintenance events. As part of the Grid2Op framework, it allows agents to make decisions about grid topology, power re-dispatch and other operations to ensure stable electricity flow and prevent overloads [6], [7].

The environment consists of 14 substations, 20 power lines and 6 generators. The key challenge is to maintain grid stability by managing real-time power flows and avoiding failures caused by overloads. Although this environment is simplified it still requires the agent to manage the grid's real-time dynamics by adjusting power flows and topology. The action space includes both discrete actions, such as switching power lines (20 actions available) and modifying grid topology (179 actions available) and continuous actions, like adjusting generator outputs to control power flow. This combination enables agents to effectively manage the grid's configuration and energy generation to maintain stability [6], [7].

The observation space provides critical real-time information about the state of the grid, including power line capacities, voltage levels and the status of generators. This allows the agent to make informed decisions about how to adjust the grid's configuration as conditions change [7].

The reward structure of the environment consists of two components:

- **The L2RPN reward**: which aims to maximise power flow.
- **The N1 reward**: which ensures that the grid adheres to the N-1 security rule. This reward ensures that there cannot be a single point of failure within the network.

These two rewards are combined into a single scalar value called the **CombinedScalarReward**, which optimises both the efficient transmission of power and the grid's ability to handle failures.

2 Choice of Algorithms (Policy Gradient Methods)

Policy gradient methods learn a parameterised policy directly, without needing to estimate action values at each state [8], [9]. This makes these methods particularly suitable for complex, high-dimensional and continuous action spaces [10], like those found in power grid management. With policy gradient methods, actions are selected stochastically based on learned policy parameters, allowing for more flexibility in making decisions compared to deterministic or value-based methods [11]. This flexibility is crucial for environments requiring real-time adjustments, such as controlling power flows. Additionally, policy gradient methods, especially those combined with value functions, can handle stochastic environments efficiently and tend to converge faster [12].

2.1 Actor-Critic

Actor-Critic combines the benefits of policy gradient methods with value function approximation [8], [13]. While REINFORCE—a basic policy gradient method that uses Monte Carlo estimates [8]—directly updates the policy using the entire return at the end of an episode, it can suffer from high variance and slow convergence due to its reliance on episodic returns [14]. Actor-Critic reduces this variance by employing a value-based critic to provide feedback at each step [15], allowing for more stable and faster learning during training [11]. Additionally, Actor-Critic has performed well in environments with both continuous and discrete actions [12], for example: The Asynchronous Advantage Actor-Critic (A3C) algorithm has been employed for autonomous voltage control in power grids [16], The Upper Confidence Bound-based Advantage Actor-Critic (UCB-A3C) algorithm has been proposed for microgrid energy management [17], Soft Actor-Critic (SAC) algorithms have been used to optimise battery dispatch in grid-connected microgrids [18], and grid dispatching strategies aimed at maximising renewable energy supply and grid stability [19], Actor-Critic neural networks have been integrated into distributed RL schemes to enhance frequency regulation in power grids, improving both short-term performance and long-term strategic utility [20]. Therefore, Actor-Critic was chosen as the algorithm taught in class for implementation of this project.

While Actor-Critic methods appear to be a reasonable candidate algorithm for the `l2rpn_case14_sandbox` environment due to their ability to balance policy improvement and real-time action evaluation, one of the major challenges faced when implementing the algorithm is setting appropriate step sizes. In Actor-Critic, large step sizes during policy updates can lead to incorrect adjustments, which are hard to recover from [12].

2.1.1 Algorithm

2.2 Proximal Policy Optimisation

2.2.1 Introduction

Introduced by [21], Proximal Policy Optimisation (PPO) is a model-free, policy gradient method that uses a 'surrogate' objective function and stochastic gradient ascent to update policies [22]–[24]. It is designed to improve the stability and sample efficiency of RL algorithms. For example; the step-size sensitivity issue of Actor-Critic methods, by limiting policy updates within a "safe" range. This ensures stable and controlled changes, which is especially important in environments with significant uncertainty and dynamic changes [12], [21] such as the `l2rpn_case14_sandbox` environment.

PPO is especially effective in continuous control tasks [25] such as power grid management, where sudden changes such as switching multiple power lines or overly aggressive power redistribution can

Algorithm 1 Actor-Critic Algorithm

Initialize policy parameters θ_π and value function parameters θ_v
while Not Converged ... **do**
  Initialize state s_0
  for each time step t **do**
    Select action a_t based on policy $\pi(a_t|s_t;\theta_\pi)$
    Execute a_t and observe reward r_t and new state s_{t+1}
    Compute TD-error: $\delta_t = r_t + \gamma V(s_{t+1};\theta_v) - V(s_t;\theta_v)$
    Update critic (value function):
    $\theta_v \leftarrow \theta_v + \alpha\delta_t \nabla_{\theta_v} V(s_t;\theta_v)$
    Update actor (policy):
    $\theta_\pi \leftarrow \theta_\pi + \beta\delta_t \nabla_{\theta_\pi} \log \pi(a_t|s_t;\theta_\pi)$ Set $s_t \leftarrow s_{t+1}$
  end for
end while

lead to network instability and trigger cascading failures if overloaded lines exceed their thermal limits [4]. It potentially mitigates this risk using its clipping mechanism by preventing large policy updates [26], [27], resulting in smoother, more controlled decision-making [26], that helps maintain grid stability and avoid destabilising actions [28].

Other successful applications of PPO includes robotic control, autonomous driving, and Unmanned Aerial Vehicle (UAV) navigation, demonstrating its versatility and robustness [29], [30]

2.2.2 Policy Gradient Methods and Large Update Problems

Policy gradient methods aim to improve a policy by adjusting its parameters in the direction that maximises expected returns. However, these methods can be unstable, especially in complex environments like power grid management. The key challenge is to update the policy effectively without making drastic changes that could lead to instability.

The standard policy gradient update can be expressed as:

$$\theta_{\text{new}} = \theta_{\text{old}} + \alpha \nabla_\theta J(\theta) \tag{1}$$

where:

- θ represents the policy parameters
- α is the learning rate
- $J(\theta)$ is the expected return
- ∇_θ denotes the gradient with respect to θ

In practice, this gradient is estimated using sampled data:

$$\nabla_\theta J(\theta) \approx \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{2}$$

where:

- $\hat{\mathbb{E}}_t$ represents the empirical average over a batch of samples
- $\pi_\theta(a_t|s_t)$ is the probability of taking action a_t in state s_t
- $\hat{A}_t$ is the advantage estimate (how much better an action is compared to average)

The challenge with this approach is that it can lead to large, potentially harmful updates [21], especially in sensitive environments like power grids where a drastic change in policy could lead to catastrophic failures [5].

2.2.3 Trust Region Methods: Improving Stability

TRPO addressed this issue by limiting the extent of policy changes. It does this by maximising a "surrogate" objective function while constraining the change in the policy:

$$\max_{\theta} \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \tag{3}$$

where:

- $\pi\theta_{\text{old}}$ represents the old policy before update
- The fraction $\frac{\pi_\theta}{\pi_{\theta_{\text{old}}}}$ is the probability ratio between new and old policies

subject to a constraint on how much the new policy can differ from the old one:

$$\hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]] \leq \delta \tag{4}$$

where:

- KL represents the Kullback-Leibler divergence
- δ is the constraint threshold for policy change

This approach ensures more stable learning but is computationally complex and difficult to implement, especially in architectures that include noise or parameter sharing [21].

2.2.4 PPO: Simplifying TRPO

PPO simplifies TRPO while maintaining its benefits. It introduces two key innovations:

Clipped Surrogate Objective: This function limits the size of policy updates:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{5}$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio
- ϵ is the clipping parameter (typically 0.2)
- $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ constrains the ratio to $[1-\epsilon, 1+\epsilon]$

This objective function encourages small, beneficial changes to the policy while preventing large, potentially harmful updates. In the context of power grid management, this means the agent can gradually learn to make better decisions without risking sudden, drastic changes that could destabilise the grid.

Adaptive KL Penalty: As an alternative or complement to clipping, PPO can use a penalty on the difference between the old and new policies, adjusting this penalty to maintain a target level of policy change.

These innovations allow PPO to achieve stable learning with simpler implementation than TRPO [21], making it particularly suitable for complex environments like power grid management in the `l2rpn_case14_sandbox`.

2.2.5 PPO Algorithm

Schulman et al. [21] present the core PPO algorithm as shown in Algorithm 2. Below is a more detailed explanation of each step:

Algorithm 2 Proximal Policy Optimisation (PPO)

for iteration = 1, 2, … **do**
  for actor = 1, 2, …, N **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T time-steps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimise surrogate L w.r.t. θ, with K epochs and mini-batch size M $\leq$ NT
  $\theta_{\text{old}} \leftarrow \theta$
end for

A more detailed explanation of the algorithm is as follows:

- The algorithm runs for multiple iterations to continuously improve the policy.
- Within each iteration, N parallel actors interact with the environment:
  - Each actor runs the current policy ($\pi_{\theta_{\text{old}}}$) in the environment for T timesteps.
  - T is typically shorter than a full episode, allowing for more frequent policy updates.
- For each timestep, the algorithm computes advantage estimates ($\hat{A}_1$ to $\hat{A}_T$):
  - These estimates represent how much better an action is compared to the average action in a given state.
- The algorithm then optimises the surrogate objective function (L):
  - This optimisation is done with respect to the policy parameters (θ).
  - It uses K epochs of mini-batch stochastic gradient descent.
  - The mini-batch size (M) is chosen to be less than or equal to the total number of timesteps collected (NT).
  - This step is where PPO differs from standard policy gradient methods, allowing multiple optimisation steps on the same data.
- Finally, the old policy parameters are updated to the new optimised values.

2.3 Main Benefits of PPO

The main benefits of PPO generally and specifically in relation to the `l2rpn_case14_sandbox` environment are:

1. **Stability**: The clipped surrogate objective prevents excessively large policy updates, ensuring more stable learning [21]. This could be particularly valuable in the `l2rpn_case14_sandbox` environment, where maintaining grid stability is crucial. Sudden changes in policy can lead to network instability, such as cascading failures triggered by exceeding thermal limits on power lines [5]. PPO helps mitigate these risks by ensuring controlled, gradual updates that prevent destabilising actions [28].
2. **Simplicity**: PPO is simpler to implement than methods like TRPO, making it computationally efficient [21].
3. **Flexibility**: PPO can handle both discrete and continuous action spaces, making it suitable for a wide range of environments [21]. This flexibility is particularly useful in the `l2rpn_case14_sandbox` environment, which involves discrete actions (such as switching power lines on/off) and continuous actions (like adjusting generator outputs). This ability to handle mixed action spaces aligns well with the environment's requirements for managing real-time grid operations [12].
4. **Sample Efficiency**: By allowing multiple optimisation steps on the same batch of data, PPO makes more efficient use of collected samples [21]. In the `l2rpn_case14_sandbox` environment, where optimising computational resources is crucial for training agents in complex, real-time power grid simulations, this efficient use of samples helps improve learning while reducing computational requirements [28].
5. **Performance**: Despite its simplicity, PPO has demonstrated superior sample efficiency and performance in continuous environments compared to traditional Actor-Critic methods [21]. This makes it a promising choice for managing dynamic environments like power grids, where both performance and stability are critical [5]. PPO's capacity to handle these demands makes it particularly well-suited for processing real-time grid state information and managing complex power flows [12].

2.4 Conclusion

Given its many advantages, PPO has been selected as one of the two algorithms for this project, along with the Actor-Critic model. PPO's stability, flexibility and proven effectiveness in continuous control tasks make it particularly well-suited for the complex, dynamic environment of the

l2rpn_case14_sandbox. Its ability to handle both discrete and continuous actions aligns well with the diverse control requirements of power grid management.

In the sections that follow, both PPO and Actor-Critic algorithms will be implemented on the l2rpn_case14_sandbox environment. Iterative improvements will also be applied for both algorithms to improve their effectiveness in managing the l2rpn_case14_sandbox environment.

3 Methodological Approach

3.1 Overview and Objectives

The primary aim of this project is to compare the performance of different RL algorithms and improvements rather than focusing solely on optimal performance. This approach takes computational limits into account and aims for stable and interpretable results from each algorithm.

Separate environments were used for training and testing, allowing agents to iteratively improve their policies through direct interaction during training while being evaluated on unseen data in testing. This separation allows for an unbiased performance comparison across all project stages.

3.2 Training Protocol

Training was limited to a maximum of 100,000 time-steps per agent, providing adequate time for policy learning while controlling computational cost. An early stopping mechanism was used, stopping training if there was less than a 5% improvement in median episode reward over a 20-episode rolling window for 20 consecutive episodes.

The median reward was chosen rather than the mean for its stability, given the high variability in rewards. Mean values can be skewed by outliers, which may not accurately reflect the agent's progress. This approach ensured training continued only when genuine improvements were observed.

To maintain consistency across iterative improvements, default hyperparameters were applied to both A2C and PPO agents, helping to avoid potential biases while focusing on adjustments to observation and action spaces. However, the discount factor was deliberately set at 0.99 for both agents to incentivise long-term rewards. Although a default, this value was specifically chosen as it aligns with PowRL's implementation, where a 0.99 discount factor contributed to effective grid stability in the L2RPN NeurIPS 2020 challenge [31].

3.3 Testing Protocol

Testing was conducted on 100 independent environments with seeds chosen to avoid overlap with training data, ensuring fair and unbiased evaluation. This setup provided consistent conditions to assess improvements across iterations. This setup ensured that performance differences were due to algorithm changes, not different environments, allowing for reliable comparisons to be made across iterations.

Due to reward volatility, the median episode reward was used as the primary evaluation metric for the trained agents in the test environments. The median better represents typical performance by reducing the effect of rare but extreme reward values that could distort the mean. Although a larger test set might have smoothed variability further, computational constraints limited this option.

4 Baseline Implementation

4.1 Setup

In the baseline iteration, agents were provided access to the full set of available actions and observations using BoxGymObsSpace and BoxGymActSpace, which convert Grid2Op's complex data into continuous Gym-compatible spaces.

4.2 Training Results

The training progress plots for A2C (Figure 1) and PPO (Figure 2) show different learning behaviours. PPO shows faster initial learning but flattens earlier, reaching early stopping in fewer episodes. A2C shows a more gradual improvement over a longer training period, with significantly higher variance in both episode rewards and survival times (note the different y-axis scales). While PPO's training curve shows more stable learning, aligning with its design limiting large policy updates, A2C's higher volatility eventually leads to better absolute performance despite less stable training.

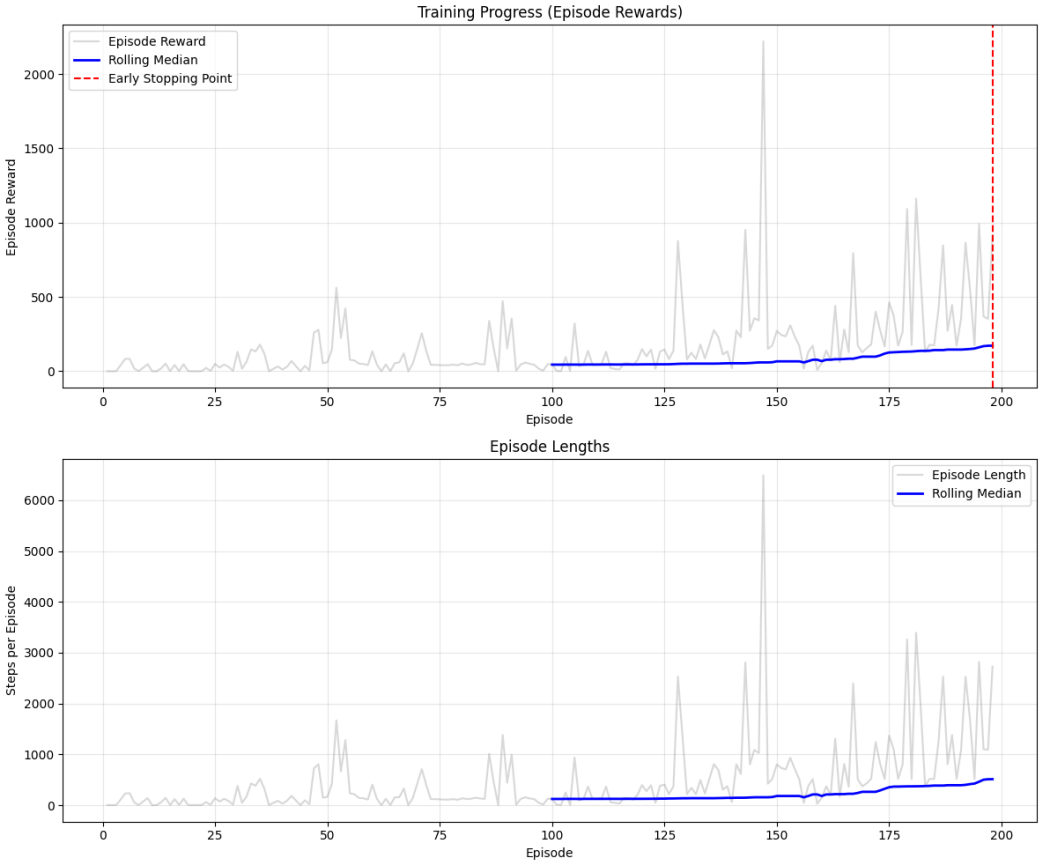

Figure 1: Training progress of the A2C algorithm until early stopping was reached. The top plot shows episode rewards, while the bottom plot presents episode lengths, each with a rolling median (blue line) over the last 20 episodes. The red dashed line indicates the early stopping point, where training was halted according to the early stopping criteria.

4.3 Test Results

To benchmark performance, two baseline agents were used: a "Do Nothing" agent that never takes actions and achieves a median reward of 0 and median survival time of 3 time-steps, and a "Random" agent that selects actions randomly that obtains median reward of 42 and median survival time of 147 time-steps.

As shown in the test episode reward distribution (Figure 3), both PPO and A2C outperform these baselines in terms of episode rewards and survival times, but with different characteristics. PPO achieves a median reward of 170 with relatively limited variance, while A2C reaches a higher median reward of 359 but lacks stability, showing much greater variability in its performance. While hyperparameter tuning could potentially improve both the performance and stability of each agent, these results using default configurations do show key differences between performance and stability of each algorithm.

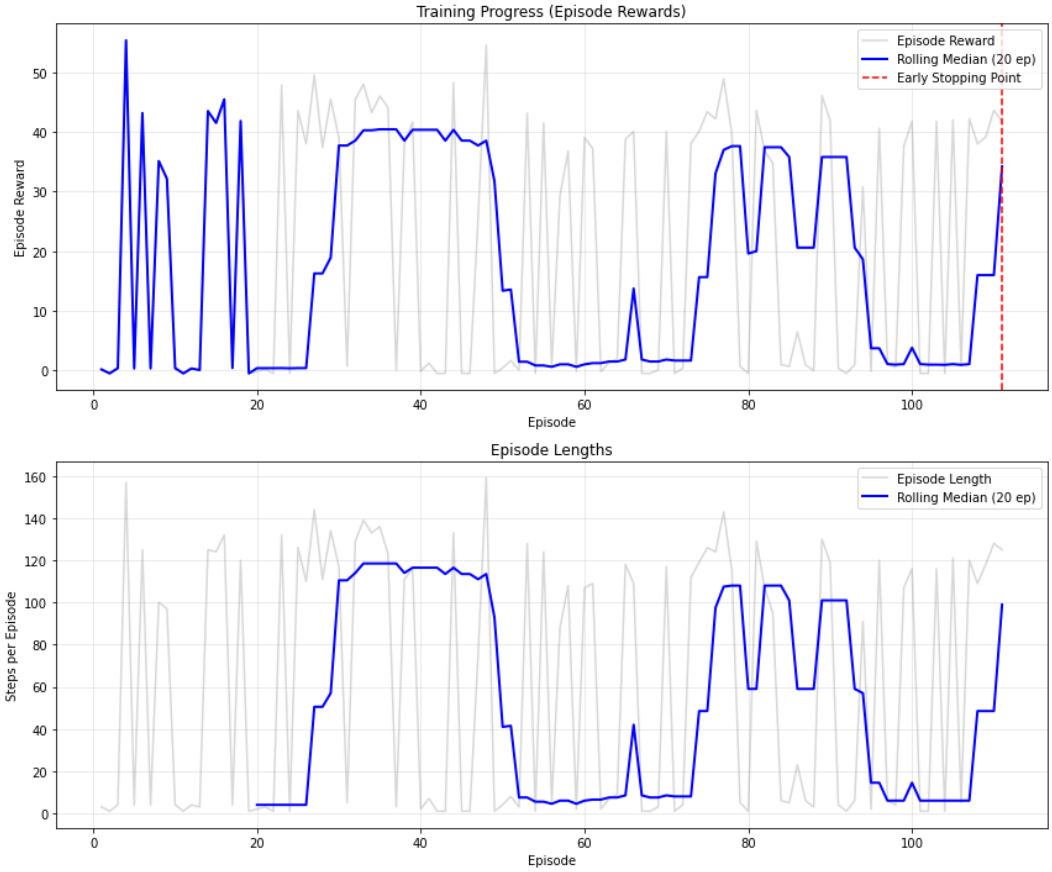

Figure 2: Training progress of the PPO algorithm until early stopping was reached. The top plot shows episode rewards, while the bottom plot presents episode lengths, each with a rolling median (blue line) over the last 20 episodes. The red dashed line indicates the early stopping point, where training was halted according to the early stopping criteria.

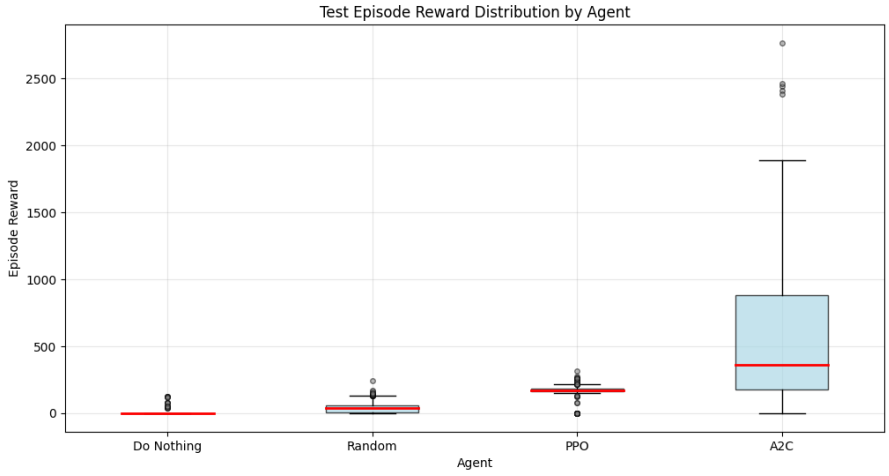

Figure 3: Distribution of test episode rewards across different agents. PPO and A2C show higher rewards compared to the Do Nothing and Random agents, with A2C achieving the highest rewards overall.

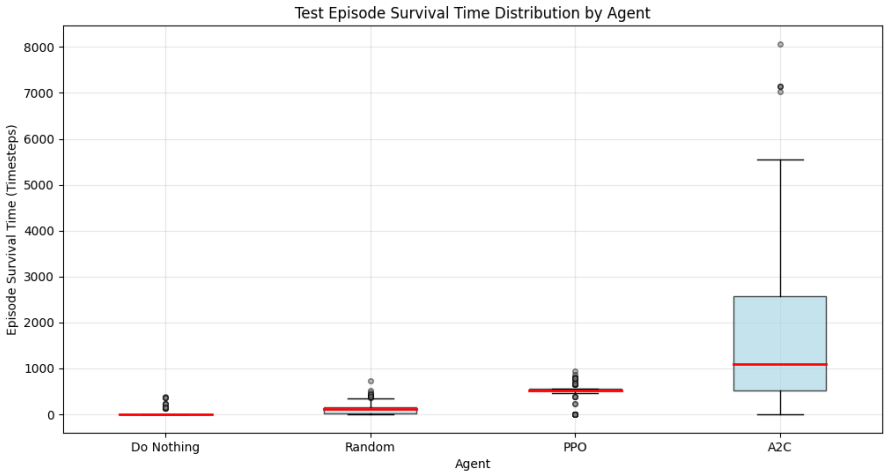

Figure 4: Distribution of test episode survival times across different agents. A2C has the longest survival times, followed by PPO, while Do Nothing and Random agents have significantly lower survival times.

5 First Improvement: Reducing Observation and Action Spaces

5.1 Setup

In order to improve upon the baseline, the observation and action spaces were reduced to focus the agent on essential information. This reduction helps agents learn with fewer, more relevant inputs [5], [31], [32].

Based on a review of the available literature, the following observations were selected as critical:

- **rho**: Represents the line load ratio, which measures each line's load against its capacity, helping prevent overloads by signalling lines nearing critical thresholds [5], [32].
- **gen_p** and **load_p**: These observations represent generation (`gen_p`) and load (`load_p`) values. Monitoring these values helps agents maintain the critical balance between supply and demand in real-time, which is foundational for grid stability [32].
- **topo_vect**: The topology vector (`topo_vect`) gives the network's structural configuration, showing how different parts of the grid are connected. This observation allows the agent to adapt to changes in grid configuration, facilitating dynamic topology adjustments that support grid stability [5], [32].
- **v_or** and **v_ex**: These observations give the voltage levels at the origin and extremity of each line. These observations were chosen to help prevent overloads and maintain power quality, indirectly based on [33].

For the action space, `set_bus` and `change_bus` were chosen as key discrete actions to reconfigure buses at substations, allowing flexible topology adjustments and power flow redirection [31], [32].

Representing these actions with `MultiDiscreteActSpace` instead of `BoxGymObsSpace` improved action precision by letting the agent select specific bus configurations. While this representation enhanced training performance, it significantly increased computational load, requiring a reduction in maximum time-steps from 100,000 to 10,000. This adjustment resulted in fewer training time-steps, giving this iteration less opportunity to learn compared to the baseline.

5.2 Training Results

The training progress plots for both A2C (Figure 5) and PPO (Figure 6) show very different patterns compared to the baseline implementation. Both algorithms have extremely short training runs of only

7 episodes before reaching early stopping conditions, with nearly identical learning patterns. This very early stopping is possibly caused by two factors: the computational overhead introduced by the MultiDiscreteActSpace representation of topology actions using bus setting and changing operations and potentially too restrictive early stopping conditions.

Ideally, with more computational resources, the maximum number of training time-steps would be increased from 10,000 to the baseline's 100,000 and early stopping patience would also be increased to allow for more thorough exploration and policy improvement. Hyperparameter tuning, particularly of learning rates and policy network parameters, could also help optimise performance.

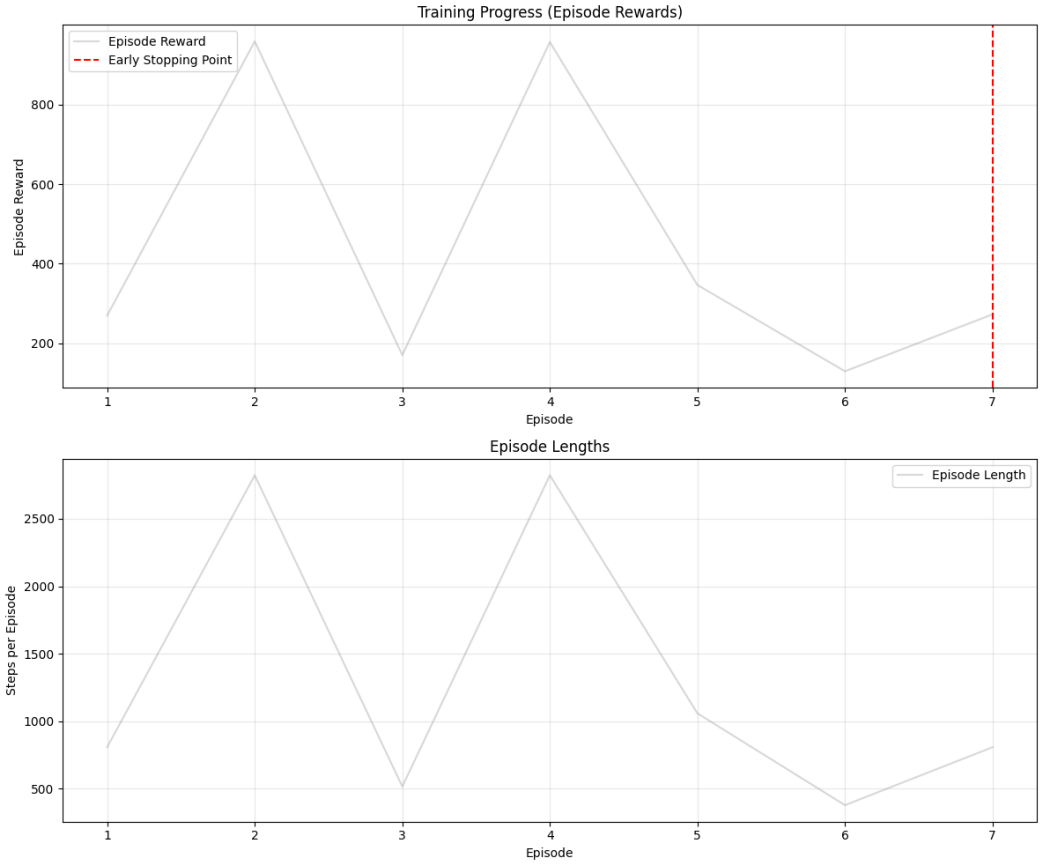

Figure 5: Training progress of A2C until early stopping was reached. The top plot shows episode rewards, while the bottom plot presents episode lengths

5.3 Test Results

The test results with reduced observation and action spaces, as shown in Figure 7, show steady improvements in median episode reward compared to the baseline implementations, despite the shortened training. PPO's median reward increased from 170 to 337, while A2C improved from 359 to 371.

The median survival times, displayed in Figure 8, also show improvement. PPO's median survival time increased from 516 to 986 time-steps, while A2C's median survival time remained similar, changing from 1092 to 1098 time-steps. Both algorithms achieved similar performance levels, with PPO's median reward now much closer to A2C's. The variance in performance also became more similar between the algorithms, with both showing similar spread in their reward distributions.

These improvements are particularly significant given both the significantly reduced training period (10,000 versus 100,000 time-steps) and the early stopping of training after only 7 episodes. This suggests that the reduced observation and action spaces, along with the more precise discrete

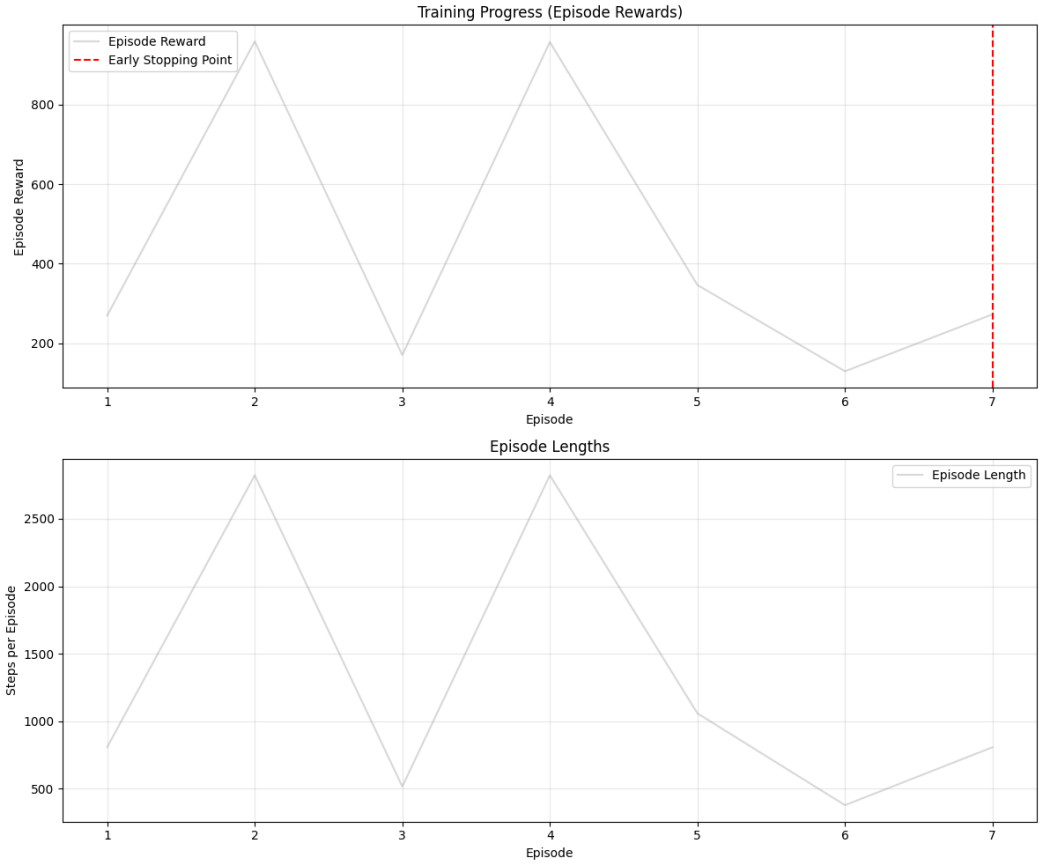

Figure 6: Training progress of the PPO algorithm until early stopping was reached. The top plot shows episode rewards, while the bottom plot presents episode lengths.

action representation, successfully focused the agents on the most relevant parts of the environment. However, the survival rates remained challenging for both algorithms, with PPO achieving only 1% survival rate and A2C showing no complete episode survivals. This shows that while the reduced spaces enabled better reward optimisation, maintaining long-term grid stability remains a significant challenge although this iteration could benefit from extended training periods and tuned hyperparameters.

6 Second Improvement: Reward Shaping

6.1 Setup

Reward shaping plays a crucial role in reinforcement learning by helping guide agents toward desired behaviors. Careful design of the reward function is crucial to effectively communicate learning objectives to the agent. This becomes particularly important for power grid management where the base environment reward may not provide sufficient learning signal to develop robust control policies [12].

In the L2RPN challenge, several approaches to reward design have been explored. [4] found that the standard competition reward alone was not ideal as a learning signal, particularly during overload scenarios and that successful teams often develop various reward schemes to better guide their agents toward desired behaviours.

In line with this, reward shaping consisting of the following four key components, was implemented:

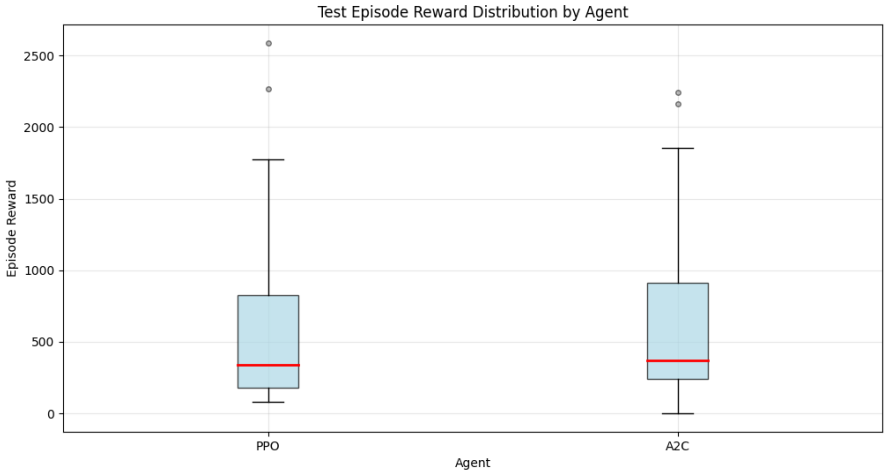

Figure 7: Distribution of test episode rewards across different agents. PPO and A2C show higher rewards compared to the Do Nothing and Random agents, with A2C achieving the highest rewards overall.

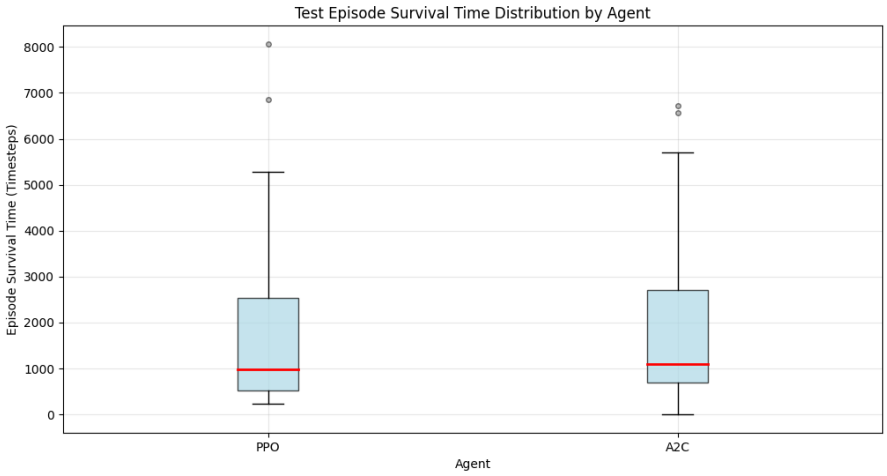

Figure 8: Distribution of test episode survival times across different agents. PPO has the longest survival times, followed by A2C, while Do Nothing and Random agents have significantly lower survival times.

- **Line Capacity Improvement**: A small bonus (0.1 × improvement) rewards reductions in maximum line capacity usage between steps. This encourages the agent to actively manage power flows to prevent overloads.
- **Sustained Improvement Bonus**: An additional small reward (0.05) is given for maintaining reduced line usage over three consecutive steps, encouraging consistent improvements in grid stability.
- **Critical Usage Penalty**: When line usage exceeds 95% capacity, a penalty (-0.1 × excess) discourages operating near critical thresholds, creating a margin of safety against potential overloads.
- **Early Game Incentive**: A small bonus (0.05) is given when the maximum line capacity stays below 80% during the first 50 time-steps of an episode, promoting safer grid conditions early on.

The shaping components were deliberately kept small compared to the environment's original reward to avoid dominating this main learning signal. This ensures that the shaped rewards guide the agent toward safer strategies while maintaining the main objective of grid stability.

Test performance was however evaluated using the original reward function to ensure fair comparison with previous iterations.

While initially intended as an improvement to be applied to the reduced observation and action space implementation, reward shaping did not improve performance of this version. However, when applied with full observation and action spaces, reward shaping showed significant improvements for the A2C algorithm, but not for PPO. Therefore, this iteration proceeded with the complete spaces from the baseline implementation.

6.2 Training Results

The training progress plots for A2C (Figure 9) and PPO (Figure 10) show very different learning patterns. A2C shows significantly more stable and consistent improvement compared to previous iterations, with a gradual progression in both total rewards and episode lengths. The training exhibits steady improvement from episodes 0-120, with initial gradual gains leading to more variable but higher performance in later episodes, occasionally exceeding rewards of 2500.

PPO's training pattern shows much shorter training time compared to A2C, reaching early stopping at around episode 110. The reward shaping components are less stable compared to A2C, suggesting that the algorithm may have had difficulty consistently using the shaped rewards.

6.3 Test Results

The test results given in Figure 11, show very different results between the two algorithms. A2C achieved its best performance across all iterations with a median reward of 458, while PPO showed regression from its previous best performance with a median reward of 160. A2C does however show considerably higher variance, indicating inconsistent policy execution despite the improved median performance.

The survival times, displayed in Figure 12, show similar divergence between the algorithms. Compared to the previous best iteration, PPO's median survival time decreased significantly from 986 time-steps to 505 time-steps, while A2C's median survival time improved from 1098 time-steps to 1377 time-steps.

7 Comparison and Discussion

As shown in Figure 13 and Table 7, both A2C and PPO significantly outperformed the baseline agents (Do Nothing or Random Agent) across all iterations, with A2C consistently achieving better performance than PPO. The final implementation using A2C with reward shaping achieved the best overall performance, with a median reward of 458.1 and median survival time of 1,376.5 time-steps.

The reward distributions given in Figure 13 and survival times in Figure 14 show distinct characteristics of the two algorithms across iterations. The reduced observation and action space implementation

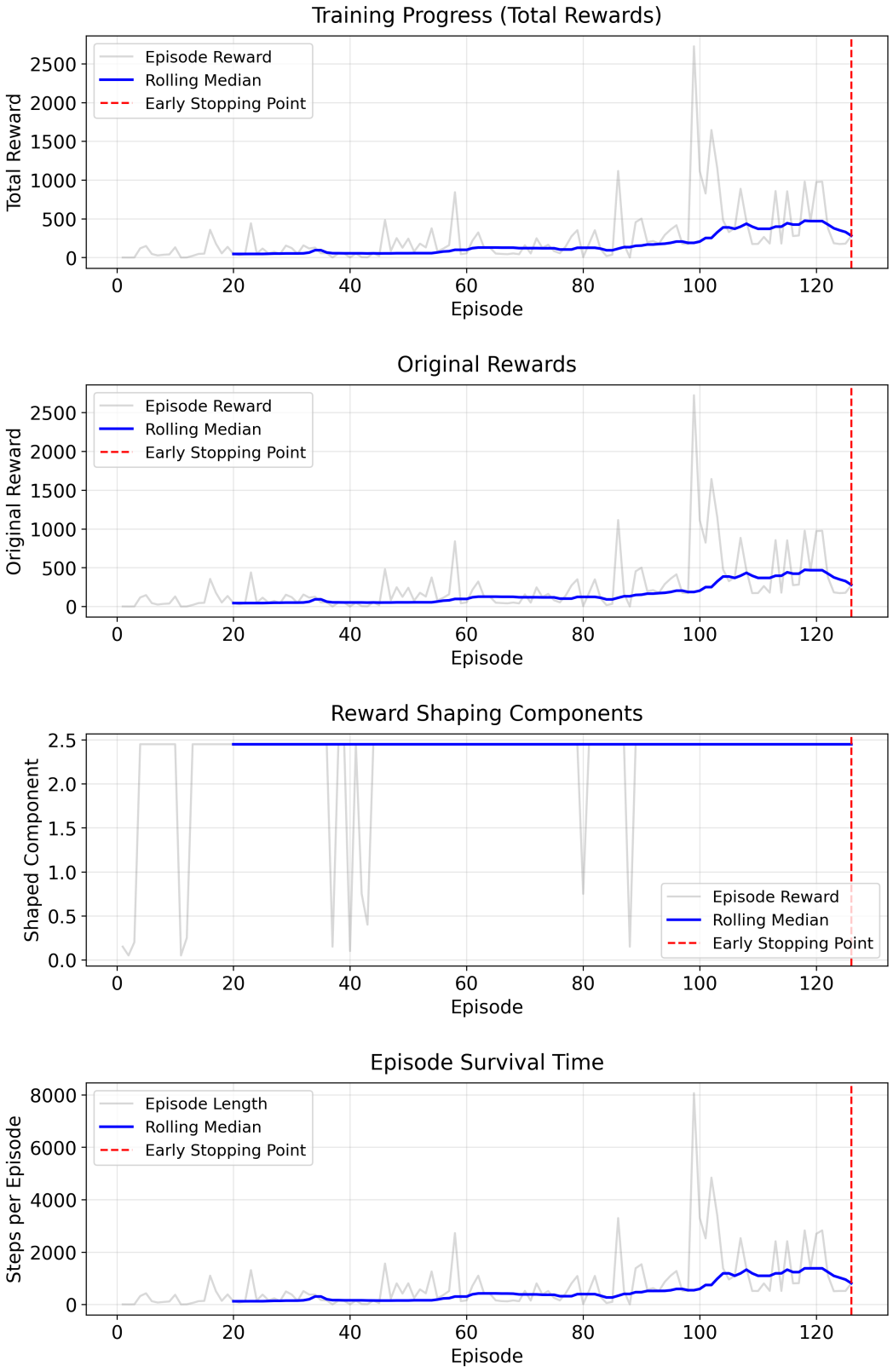

Figure 9: Training progress of the A2C algorithm until early stopping was reached. The top plot shows episode rewards, while the bottom plot presents episode lengths, each with a rolling median (blue line) over the last 20 episodes. The red dashed line indicates the early stopping point, where training was halted according to the early stopping criteria.

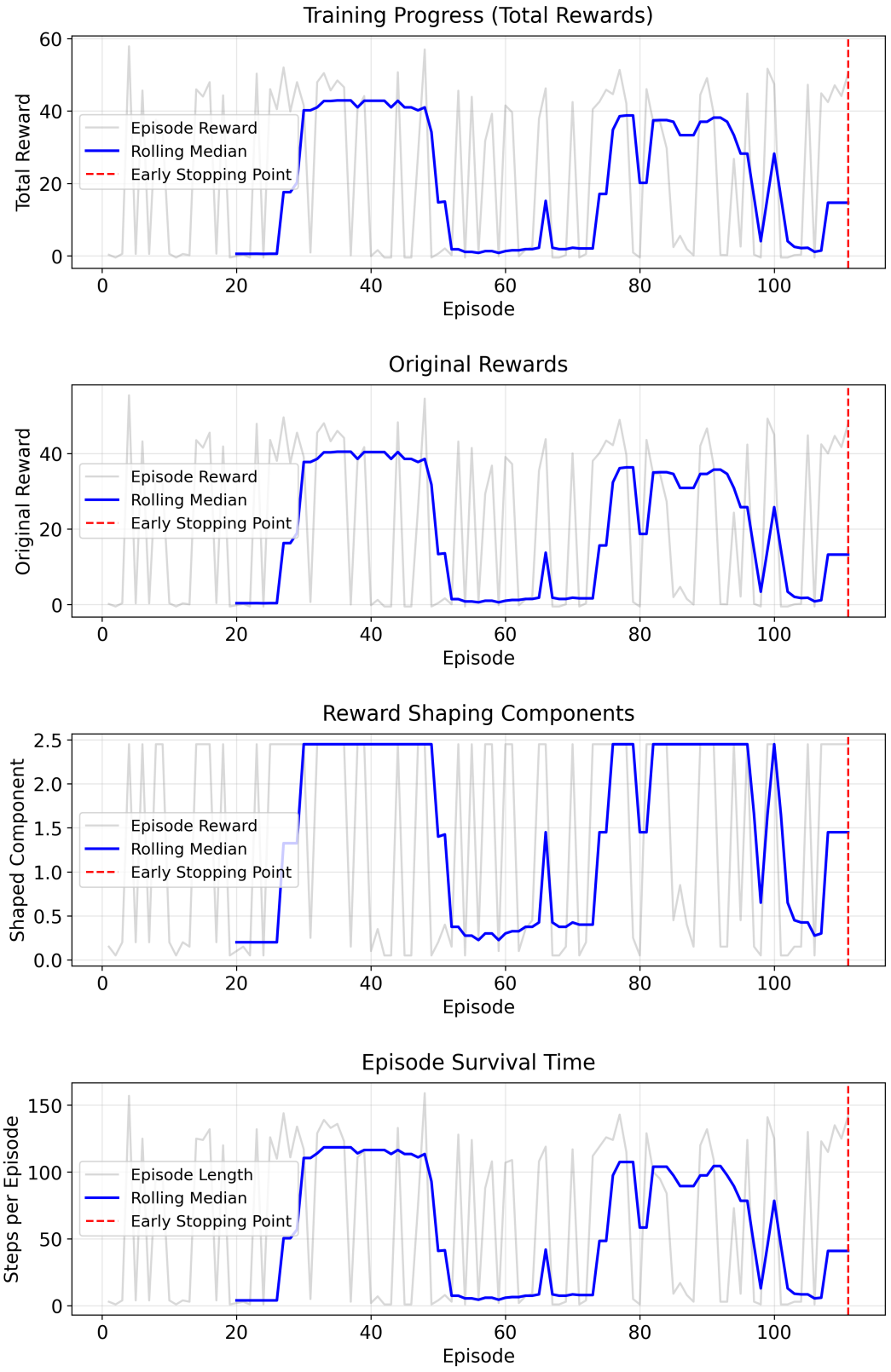

Figure 10: Training progress of the PPO algorithm until early stopping was reached. The top plot shows episode rewards, while the bottom plot presents episode lengths, each with a rolling median (blue line) over the last 20 episodes. The red dashed line indicates the early stopping point, where training was halted according to the early stopping criteria.

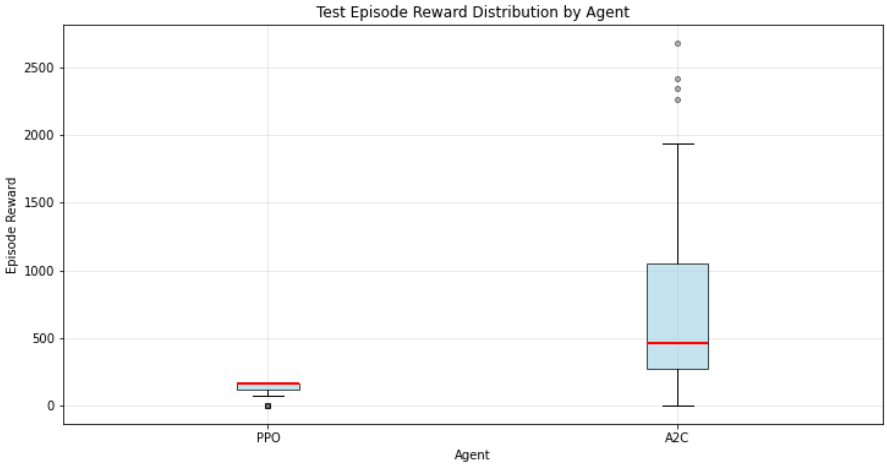

Figure 11: Distribution of test episode rewards across different agents. PPO and A2C show higher rewards compared to the Do Nothing and Random agents, with A2C achieving the highest rewards overall.

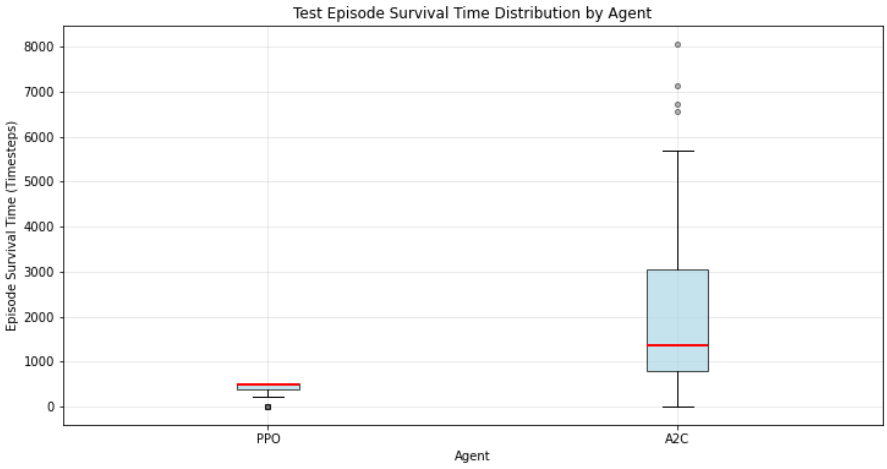

Figure 12: Distribution of test episode survival times across different agents. PPO has the longest survival times, followed by A2C, while Do Nothing and Random agents have significantly lower survival times.

improved both algorithms' performance compared to their baselines, with PPO showing particularly good improvement (median reward increasing from 170.3 to 336.7, as shown in Table 7). This suggests that focused state and action representations can help agents learn more effectively, even with reduced training time. However, the reward shaping iteration produced differing results between the algorithms. While A2C's performance improved greatly to achieve its highest median reward of 458.1, PPO's performance decreased, with its median reward dropping to 160.2.

A consistent pattern across Figures 13 and 14 was the trade-off between performance and stability. While A2C achieved higher median rewards and survival times, it showed significantly greater variance than PPO across all iterations, as sown by the larger inter-quartile ranges. While this variance could be a concern for real-world scenarios, it's important to note that no hyperparameter tuning was performed to address variability for either algorithm.

Agent	**Median Reward**	**Median Survival Time**
Do Nothing Agent	0.1	3.0
Random Agent	42.0	122.5
Baseline - PPO	170.3	516.0
Baseline - A2C	359.4	1092.0
Reduced Observation and Action Space - PPO	336.7	986.0
Reduced Observation and Action Space - A2C	371.2	1098.5
Reward Shaping - PPO	160.2	505.0
Reward Shaping - A2C	458.1	1376.5

Table 1: Median performance metrics across agents and iterations. The results show the superiority of RL agents over baseline approaches, with A2C consistently outperforming PPO. The reduced observation and action space improved both agents' performance compared to their baselines. Reward shaping led to mixed results - decreasing PPO's performance but significantly improving A2C's performance, leading to the best overall performance reward.

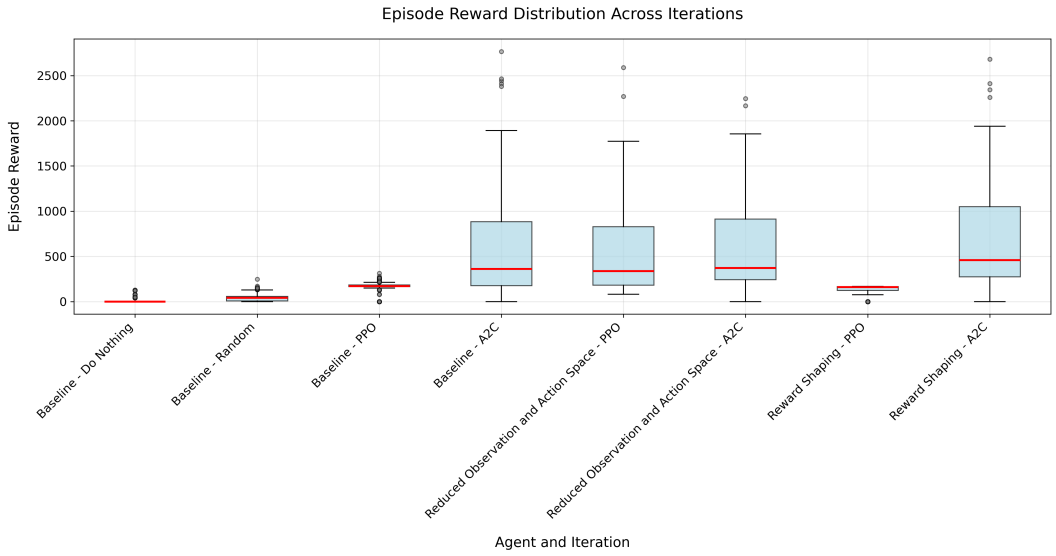

Figure 13: Distribution of episode rewards across different agents and iterations. A2C consistently outperformed PPO across all iterations, with reward shaping providing the best improvement. Both RL agents significantly outperformed the baseline Do Nothing and Random agents.

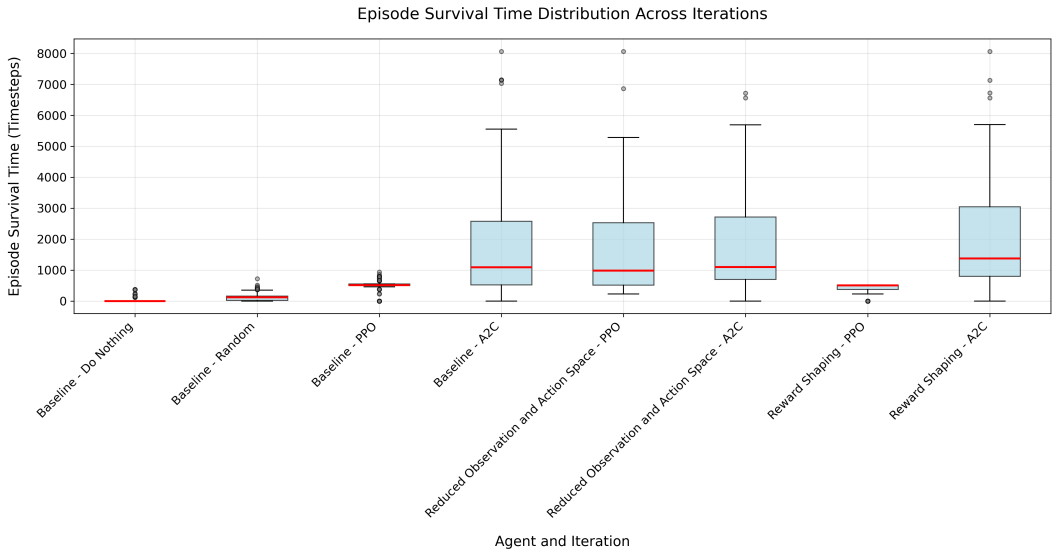

Figure 14: Distribution of episode survival times across different agents and iterations. The survival times mirror the reward distributions, with A2C achieving longer survival times than PPO across all iterations.

8 Conclusion

This project explored RL techniques, particularly A2C and PPO, within the context of the L2RPN environment for power grid management. Through multiple iterations and improvements, including reducing observation and action spaces and applying reward shaping, we demonstrated the importance of focusing agent learning on relevant information while guiding behaviors through tailored reward structures. While PPO showed initial stability, A2C ultimately outperformed it in later iterations, benefiting from the applied enhancements. Future work could include further hyperparameter tuning, extended training periods, and exploration of additional RL techniques to enhance performance and grid stability in real-time power management.

9 References

[1] H. Shakouri, S. Pandey, F. Rahmatian, and E. Paaso, "Does the increased electricity consumption (provided by capacity expansion and/or reliability improvement) cause economic growth?" *Energy Policy*, vol. 3, 2023.

[2] P. Burke, D. Stern, and S. Bruns, "The impact of electricity on economic development: A macroeconomic perspective," *International Review of Environmental and Resource Economics*, 2018.

[3] A. Kelly, A. O'Sullivan, P. de Mars, and A. Marot, "Reinforcement learning for electricity network operation," *arXiv preprint arXiv:2003.07339*, 2020.

[4] A. Marot, B. Donnot, G. Dulac-Arnold, *et al.*, "Learning to run a power network challenge: A retrospective analysis," in *NeurIPS 2020 Competition and Demonstration Track*, PMLR, 2021, pp. 112–132.

[5] A. Marot, I. Guyon, B. Donnot, *et al.*, "L2rpn: Learning to run a power network in a sustainable world neurips2020 challenge design," *Réseau de Transport d'Électricité, Paris, France, White Paper*, 2020.

[6] B. Donnot, *Grid2op- A testbed platform to model sequential decision making in power systems.* `https://GitHub.com/rte-france/grid2op`, 2020.

[7] F. Najar, "Possibilities of a decentralized power grid management in real time using multi-agent reinforcement learning," Master's thesis or report, Institut de Mathématiques d'Orsay, Université Paris-Saclay.

[8] R. Huang, T. Yu, Z. Ding, and S. Zhang, *Policy gradient*. 2020, pp. 161–212, Cited by: 7. DOI: `10.1007/978-981-15-4095-0_5`. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85089633444&doi=10.1007%2f978-981-15-4095-0_5&partnerID=40&md5=a8d64bc4b7b7488cd85c7cd7b69208f4`.

[9] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "On the theory of policy gradient methods: Optimality, approximation, and distribution shift," *Journal of Machine Learning Research*, vol. 22, 2021, Cited by: 186. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85107297227&partnerID=40&md5=f3dbcfebbbc8e88f3cf619edbca4adad`.

[10] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," Cited by: 694, 2016. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85083954383&partnerID=40&md5=c91337b04b0fa7485f8f4732060cac54`.

[11] R. Sutton and A. Barto, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[12] E. O. Arwa and K. A. Folly, "Reinforcement learning techniques for optimal power control in grid-connected microgrids: A comprehensive review," *Ieee Access*, vol. 8, pp. 208 992–209 007, 2020.

[13] J. Wen, S. Kumar, R. Gummadi, and D. Schuurmans, "Characterizing the gap between actor-critic and policy gradient," Cited by: 5, vol. 139, 2021, pp. 11 101–11 111. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85146766560&partnerID=40&md5=edcf7c99a8c1c3b6c4ff33214ff9f7a9`.

[14] M. Ghavamzadeh and Y. Engel, "Bayesian policy gradient algorithms," Cited by: 9, 2006, pp. 457–464. [Online]. Available: `https://www.scopus.com/inward/record.uri?eid=2-s2.0-85093445500&partnerID=40&md5=3d15e2fae099d9fbbc0e6361e1aa46db`.

[15] K. Van Den Houten, E. Van Krieken, and B. Heidergott, "Analysis of measure-valued derivatives in a reinforcement learning actor-critic framework," Cited by: 0; All Open Access, Green Open Access, vol. 2022-December, 2022, pp. 2736–2747. DOI: 10.1109/WSC57314.2022.10015323. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85147415341&doi=10.1109%2fWSC57314.2022.10015323&partnerID=40&md5=eb765cfadd5d717b4bee6d1129047ddc.

[16] Z. Xu, Y. Zan, C. Xu, *et al.*, "Accelerated drl agent for autonomous voltage control using asynchronous advantage actor-critic," Cited by: 6, vol. 2020-August, 2020. DOI: 10.1109/PESGM41954.2020.9281768. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85099118006&doi=10.1109%2fPESGM41954.2020.9281768&partnerID=40&md5=f99450ca807cf270cfbad32a558d77da.

[17] Y. Yang, H. Li, B. Shen, W. Pei, and D. Peng, "Microgrid energy management strategy base on ucb-a3c learning," *Frontiers in Energy Research*, vol. 10, 2022, Cited by: 4; All Open Access, Gold Open Access. DOI: 10.3389/fenrg.2022.858895. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85128459519&doi=10.3389%2ffenrg.2022.858895&partnerID=40&md5=f9c35d03155e853a50ca73facf89904d.

[18] M. Sage, M. Staniszewski, and Y. F. Zhao, "Economic battery storage dispatch with deep reinforcement learning from rule-based demonstrations," Cited by: 1, 2023. DOI: 10.1109/ICCAD57653.2023.10152299. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85164170033&doi=10.1109%2fICCAD57653.2023.10152299&partnerID=40&md5=a91111330e50cf2f0f32b51df5eddbac.

[19] L. Zhao, F. Li, Y. Zhou, and W. Fan, "Soft actor-critic-based grid dispatching with distributed training," Cited by: 0, 2023, pp. 1–6. DOI: 10.1109/MICCIS58901.2023.00007. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85173680872&doi=10.1109%2fMICCIS58901.2023.00007&partnerID=40&md5=bba65d0101d54cdbc8e11e4d04ba0d49.

[20] J. Sun, Z. Zhu, H. Li, *et al.*, "An integrated critic-actor neural network for reinforcement learning with application of ders control in grid frequency regulation," *International Journal of Electrical Power and Energy Systems*, vol. 111, pp. 286–299, 2019, Cited by: 30. DOI: 10.1016/j.ijepes.2019.04.011. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85064313580&doi=10.1016%2fj.ijepes.2019.04.011&partnerID=40&md5=c71f59e8e1d3943163a6a7214bf004d1.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[22] J. Zhou, J. Feng, and Z. Chang, "Improving proximal policy optimization with hellinger divergence," Cited by: 0, 2024, pp. 1–7. DOI: 10.1109/AIPMV62663.2024.10692111. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85206804724&doi=10.1109%2fAIPMV62663.2024.10692111&partnerID=40&md5=550c1d070bb45114c8e29276761a0805.

[23] T. Balasuntharam, H. Davoudi, and M. Ebrahimi, "Preferential proximal policy optimization," Cited by: 0, 2023, pp. 293–300. DOI: 10.1109/ICMLA58977.2023.00048. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85190145416&doi=10.1109%2fICMLA58977.2023.00048&partnerID=40&md5=97e6bbccf3891cac18c832b19947c0c0.

[24] Y. Gu, Y. Cheng, C. L. P. Chen, and X. Wang, "Proximal policy optimization with policy feedback," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4600–4610, 2022, Cited by: 33. DOI: 10.1109/TSMC.2021.3098451. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85112651578&doi=10.1109%2fTSMC.2021.3098451&partnerID=40&md5=1b197a2121578936285a937c8cf81094.

[25] J. Zhang, Z. Zhang, S. Han, and S. Lü, "Proximal policy optimization via enhanced exploration efficiency," *Information Sciences*, vol. 609, pp. 750–765, 2022, Cited by: 17; All Open Access, Green Open Access. DOI: 10.1016/j.ins.2022.07.111. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85134893495&doi=10.1016%2fj.ins.2022.07.111&partnerID=40&md5=223ed6e4a0d1a8e8c760566eacc082b0.

[26] M. Farsang and L. Szegletes, "Decaying clipping range in proximal policy optimization," Cited by: 2; All Open Access, Green Open Access, 2021, pp. 521–526. DOI: 10.1109/SACI51354.2021.9465602. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85113901970&doi=10.1109%2fSACI51354.2021.9465602&partnerID=40&md5=484fdfb727b445b61c68f142d33e8cde.

[27] H. Xu, Z. Yan, J. Xuan, G. Zhang, and J. Lu, "Improving proximal policy optimization with alpha divergence," *Neurocomputing*, vol. 534, pp. 94–105, 2023, Cited by: 6. DOI: 10.1016/j.neucom.2023.02.008. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85150053469&doi=10.1016%2fj.neucom.2023.02.008&partnerID=40&md5=0d5baa64c7d10dd2cc3a4b3835bea765.

[28] Y. Zhou, B. Zhang, C. Xu, *et al.*, "A data-driven method for fast ac optimal power flow solutions via deep reinforcement learning," *Journal of Modern Power Systems and Clean Energy*, vol. 8, no. 6, pp. 1128–1139, 2020.

[29] S. Wu, W. Xue, H. Ye, and S. Li, "A novel proximal policy optimization control strategy for unmanned surface vehicle," Cited by: 0, 2023, pp. 2815–2819. DOI: 10.1109/CCDC58219.2023.10326629. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85181799062&doi=10.1109%2fCCDC58219.2023.10326629&partnerID=40&md5=6adeaf28261ceeb8f7c72e1451979504.

[30] C. Xu, R. Zhu, and D. Yang, "Karting racing: A revisit to ppo and sac algorithm," Cited by: 4, 2021, pp. 310–316. DOI: 10.1109/CISAI54367.2021.00066. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85127371056&doi=10.1109%2fCISAI54367.2021.00066&partnerID=40&md5=b84cec8de2bd5a09dee893342a48e663.

[31] A. Chauhan, M. Baranwal, and A. Basumatary, "Powrl: A reinforcement learning framework for robust management of power networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 14 757–14 764.

[32] M. Lehna, J. Viebahn, A. Marot, S. Tomforde, and C. Scholz, "Managing power grids through topology actions: A comparative study between advanced rule-based and reinforcement learning agents," *Energy and AI*, vol. 14, p. 100 276, 2023.

[33] B. Zhou, H. Zeng, Y. Liu, K. Li, F. Wang, and H. Tian, "Action set based policy optimization for safe power grid management," in *Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part V 21*, Springer, 2021, pp. 168–181.