Reinforcement Learning Project

2332155 Tao Yuan, 2424161 Shakeel Malagas, Tumi Jourdan 2180153, Dean Solomon 2347848

October 27, 2024

Abstract

This study explores two reinforcement learning approaches for power grid management: Actor-Critic (A2C) and Proximal Policy Optimization (PPO). We implement and compare various improvements to both architectures, including observation space normalization, curriculum learning, and Graph Neural Networks (GNN). The A2C implementation showed significant improvements through curriculum learning, achieving more stable training and higher rewards, particularly when combined with GNN and momentum-based optimization. The PPO architecture was enhanced through a multi-agent system and hierarchical approach, though results varied across implementations. Our findings demonstrate that curriculum learning in A2C provides more stable convergence around 600 reward points, while GNN with momentum occasionally achieved rewards up to 800.

1 Introduction

Power networks form the backbone of modern infrastructure, supporting nearly every aspect of daily life through the continuous supply of electricity. From homes to industries, maintaining a stable and reliable power grid is essential to ensuring smooth operations. However, managing these grids presents significant challenges due to their complexity and the potential for disruptions caused by fluctuating demand, equipment failures, or external factors.

In this task, we focus on developing two Reinforcement Learning agents that operates within the power grid, tasked with maintaining a reliable flow of electricity while adhering to operational constraints. The agent must respond to disturbances and optimize the grid's performance to prevent blackouts or power shortages. This report outlines the design, implementation, and evaluation of the agents, as well as the insights gained throughout the process.

The source code for the project can be found at this link: https://github.com/TumiJourdan/RL-Ass-1/tree/main/Assignment

2 Actor Critic Architecture

For the algorithm covered in class we chose to use the A2C algorithm which is a deterministic version of Asynchronous Advantage Actor Critic(A3C). We chose to use this method as it can use a dictionary for observations and multi-discrete actions.
For the baseline we ran the A2C algorithm from Stable Baselines3 on the full observation space of Grid2Op.Observation.CompleteObservation and a reduced action space of only using 6 actions (one_line_set, one_sub_set, set_bus, set_line_status, sub_set_bus, redispatch).

This base model ran for 211 minutes, 176500 timesteps and had a mean reward of mainly between 420 and 570 with an early spike in training that went to 830, however, performance fell drastically afterwards before improving slowly. Overall, this method proved to have high variance in the returns and trained incredibly slowly.

2.1 Improvement 1: Normalization

For our first improvement we kept the reduced action space while reducing the observation space to 7 observations (rho, line_status, topo_vect, gen_p, load_p, p_or, p_ex) and normalized the observations.

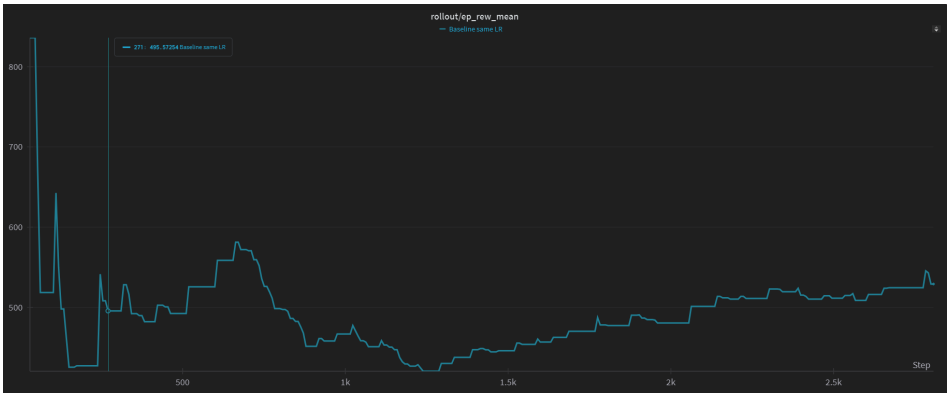

Figure 1: Average reward returned for baseline agent

Normalization of the observations ensures that values are in a manageable range and that outliers do not affect the model disproportionately. This ran for 360 minutes, 473000 timesteps and had a mean reward of between 440 and 540 while also having a spike in reward early in training to 824 which also fell. When compared to the base agent, this method ran 168% more timesteps in 71% more time showing that it ran significantly faster. However, this method was still unstable in training showcasing high variance in the average reward returned.

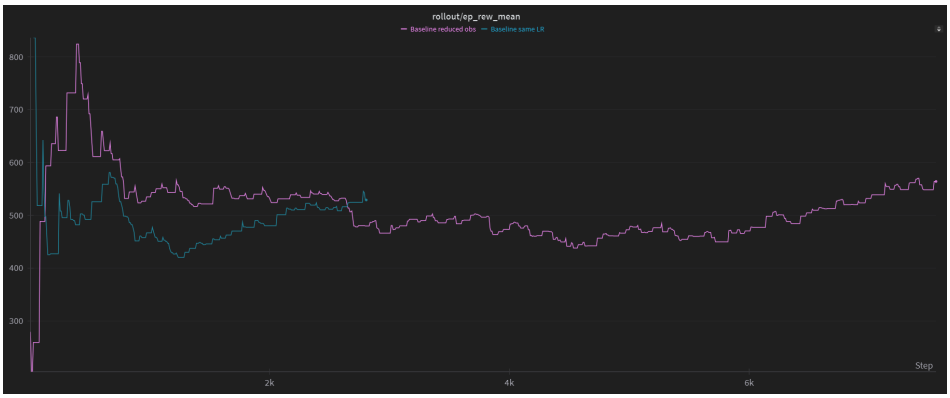

Figure 2: Average reward returned for agent with reduced observation and normalisation

2.2 Improvement 2: Curriculum Learning

For our second improvement we added curriculum learning [BLCW09] to the previous model.

Curriculum learning is when you train a model on progressively harder examples. We did this by adding actions to our agent over time. Starting at only one action and ending with 6 actions available to the agent.

The agent received a new action every 20000 timesteps until it had all 6 and then it continued to train with these till the end of the run. The idea is that the agent learns how to best utilise one action before being introduced to another one. This is especially useful for harder tasks such as this one of managing a power grid.

The first action it learns is to set a line which is a fundamental part of managing a grid. New actions allow the agent to build upon this knowledge in increments and should result in more stable training.

This improvement resulted in a run of 81 minutes, 500000 timesteps and demonstrated a much more linear growth of rewards until a reward of 563 was reached where after the reward returned became more oscillatory in nature. The reward remained relatively high and ended the run with a final average reward of 544.

This method ran 27000 more timesteps in just 23% of the time when compared to the previous model with only observation space improvements. It also provided much more stable training with no large spikes or deviances in the rewards graph.

This model was graphed in Figure 3

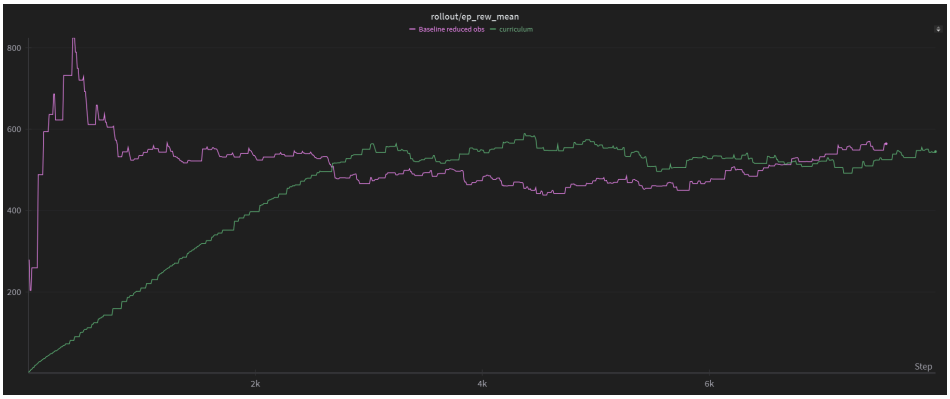

Figure 3: Average reward returned for curriculum learning agent

2.3 Improvement 3: Graph Neural Network

The approach of fully linearising the observation space and feeding it into a fully connected layer to both the actor and critic removes important structural information, and must learn it from the ground up. Taking a similar approach to CNNs the Graph Convolutional Network uses one neighbourhood kernels to intelligently learn the sub-graphs of the problem space. This embeds the graphs structure in how the network learns.

The traditional Graph Convolutional Network only uses node data, to incorporate the edge information we use the NNConv proposed by [GSR+17].

2.4 Implementation

The GNN is added as a feature extractor to the network, which then feeds its output nodes to the actor and critic networks in the A2C model (figure 4).

2.4.1 Results

The First run in Figure 5 GNN0 was promising, but on on consecutive runs the model would enter local minima strategies. To solve this we add Momentum.

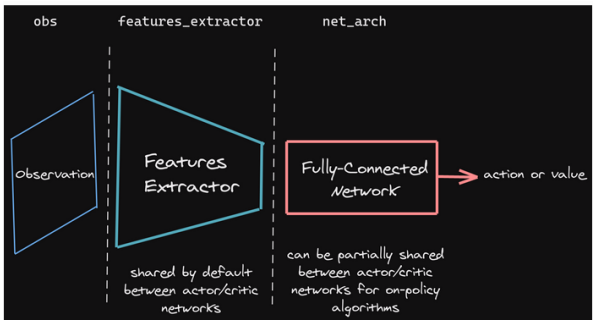

Figure 4: Feature Extraction https://stable-baselines3.readthedocs.io/en/master/guide/custom_policy.html

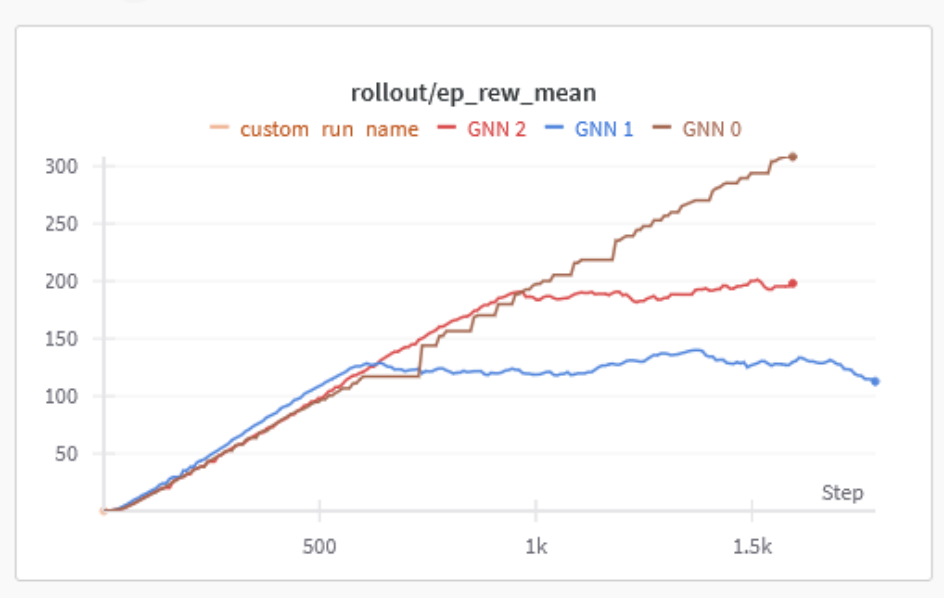

Figure 5: Average reward returned for curriculum learning agent

2.5 Other: Momentum

Momentum smooths the weight updates by accumulating past gradients, making it easier to "roll over" small local minima or dampen oscillations. Instead of relying solely on the gradient at the current step, the momentum term maintains a running sum of the previous gradients. This running sum acts like inertia, ensuring that updates in consistent directions become stronger, while oscillations cancel out.

One could imagine a ball rolling over the problem space, and adding momentum allows the ball to escape local minima.

This change improved the rewards buy upper bounding previous graphs from the onset, but is also able to converge to a higher reward of 800. As seen in Figure 6

A drawback to this is that, on a micro level, the granular data is much more noisy, leading to more varied moment to moment behaviour but a far more consistent convergence.

The maximum reward for the trained agents is much larger as seen in Fig 6 of 800.

2.5.1 Removal of normalization

When reimplementing the normalization improvement into the GNN, the results were far worse. This is due to a loss of information in the sub graphs (that the GNN relies on) that the normalization removes. Figure 7 demonstrates 3 training runs of the normalization that converge below a reward of 400.

2.6 Curriculum and GNN

2.7 Time

The GNN runs for a total of 164 minutes for 500k steps, whilst the curriculum learning runs for 81 minutes. Therefore running takes much longer due to adding a second network, which is then trained alongside the backbone actor critic network.

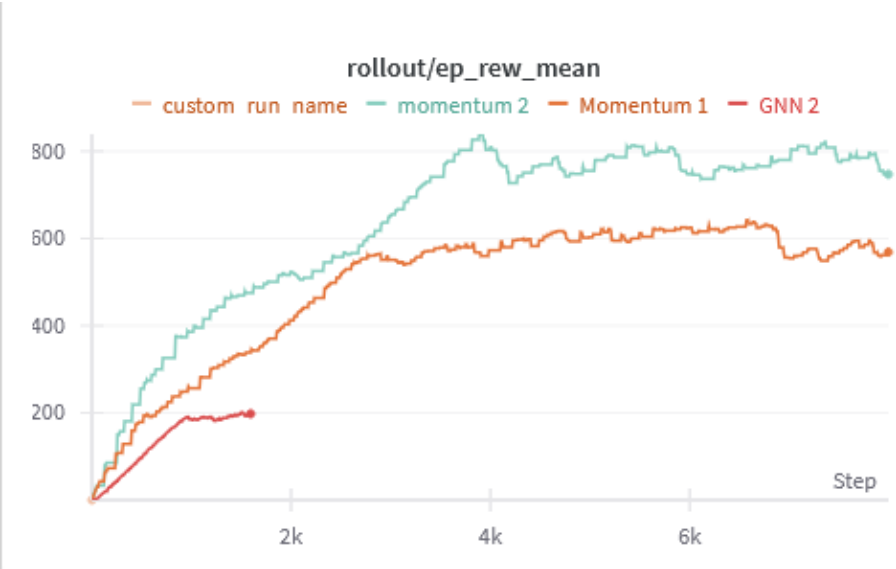

Figure 6: Average reward returned for curriculum learning agent

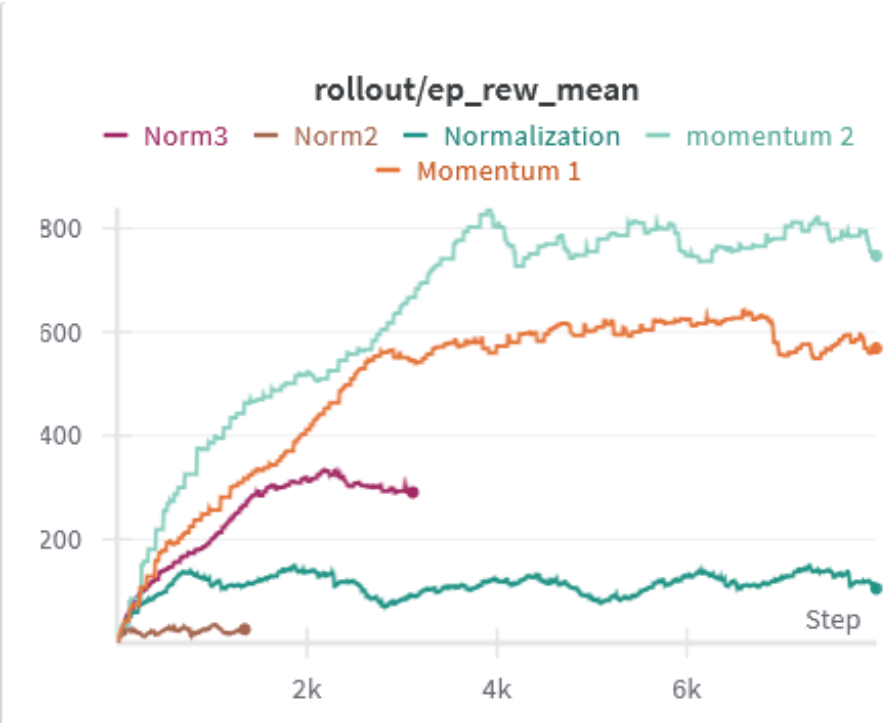

Figure 7: Average reward returned for curriculum learning agent

2.7.1 Rewards

Implementing the GNN and Curriculum together is promising sub 4k steps during training , reaching an average reward of 650 , but then post 6k steps its performance converges to 600, a similar result to the previous models, (shown in Figure 8, where the white line is the GNN + Curriculum).

In conclusion the curriculum provided a consistent convergence to a reward of 600 over many runs, whilst the GNN + Momentum could occasionally reach 800 if not caught by a local minima. Combined the GNN allowed to the Curriculum to initially train to a much higher point, although at the cost of much slower training.

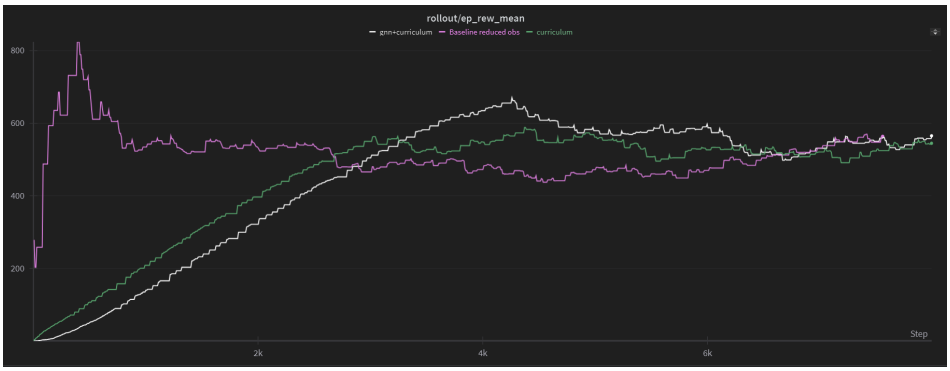

Figure 8: Average reward returned for curriculum learning agent

3 PPO Architecture

For the architecture not covered in the course, we were looking for something that would be easy to implement (given the short time frame) and had good comparability with the structure of the actions and observation spaces of the Grid2Op environment. Proximal policy optimization (PPO) met these requirements and was selected due to its ability to perform in high dimensional spaces and in environments that require resource management and strategic planning.

For all our agents we kept the reward returned as default.

For the base agent of PPO we transformed the action space into Multi-Dimensional Discrete. This is due to the limitations of PPO which allows for a dictionary for observation space but not for action space.

Figure 9: Average reward and episode length during training of base agent

For our improvements we looked at ways to minimize the loss of information from continuous action spaces (in particular for curtail and redispatch).

3.1 Improvement 1: Altering action and observation space

As with the A2C implementation the first improvement aimed at reducing the size of the problem by reducing the action and observation as well as normalizing the observations.

Action space: We noticed that in the 6 given actions, 2 were redundant for similar purposes (change vs set). In particular: set_bus : change_bus set_line_status : change_line_status redispatch curtail

Implementing both change and set for the same action would result in invalid actions. Hence it was determined it was necessary to remove the redundancies of using set. This allowed the agent to learn better as there is a smaller action space, and prevents conflicting actions.

Thus the final actions kept were

- change_bus
- change_line_status
- redispatch
- curtail

Observation space: We determined that the large amount of observations would contribute to a significant amount towards slowing of training and affecting its ability to learn.

Out of the 56 observations from Grid2Op.Observation.CompleteObservation, the 6 observations kept are

- **rho** - represents the power flow between nodes in the grid. Understanding how power is distributed across the network is essential for managing grid stability, ensuring balance between supply and demand, and avoiding overload conditions
- **topo_vect** - represents the topology vector of the grid, which encodes the current configuration of connections between grid components (e.g., buses, lines). It is crucial to know the network structure at any given time to manage the flow of electricity effectively.
- **gen_p** - captures the real power output of generators. Managing generator output is critical for balancing the grid in real time, ensuring that power generation matches consumption and preventing failures due to over- or under-generation.
- **load_p** - measures the active power consumption of loads (customers or regions). Understanding the real-time load is fundamental for balancing generation and consumption in the grid, as mismatches can cause instability or power outages.
- **actual_dispatch** - represents the actual power dispatched from generators. Knowing how much power is currently being sent into the grid is key to ensuring that the planned dispatch aligns with real-time operations and that the system stays balanced.
- **target_dispatch** - the target or planned dispatch for generators, representing the ideal power levels set by the grid operator. It is essential to compare this with the actual dispatch to identify any discrepancies and take corrective actions.

These were viewed as the most critical observations that were needed to provide the bare minimum for the agent to function. This was concluded after careful deliberation and research on similar PPO implementations done by others on github.

Of these 6 observations, we normalized **load_p** and **gen_p**. As the values returned were in the order of thousands, we normalized them to between 0 and 1. This was done as recommended in the getting started pages.

It can be noted that the graph for this improved agent looks to the exact same as the base agent. This is likely due to the agent trying to take invalid actions due to failure to adequately learning the environment. This indicates that the observations given may not be sufficient enough for the agent to learn the appropriate features. Another potential reason is that it is extremely difficult for a single agent to learn the full dynamics of the action space and the potential loss of information during discretizing of parts of the action space.

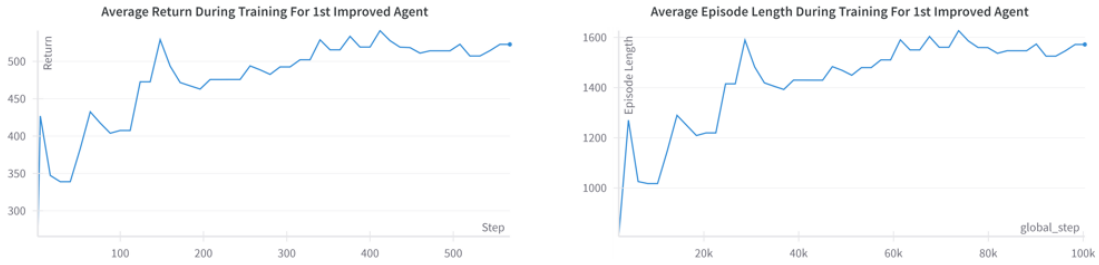

Figure 10: Average reward and episode length during training of first improved agent

3.2 Improvement 2: Multi agent system

Another way to we looked at minimizing loss of information is a divide and conquer approach where we created an agent for each of the 4 actions. Each of these agents were given the same subset of the observation space as the base agent. This allowed each agent specialize in its own action given an observation.

Furthermore, two of the four actions (curtail and redispatch) had continuous spaces. This approach allowed us to keep them as continuous spaces and thus not lose any information in discretizing them.

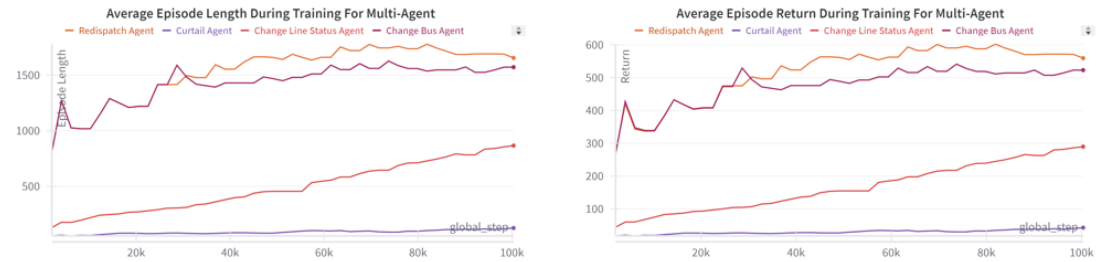

Figure 11: Average reward and episode length during training of each of the Multi Agents

Here we can notice that the change bus agent shows a trend that is highly similar to the previous agents. This may indicate that the change_bus action dominated the actions taken by the previous agents.

3.3 Improvement 3: Hierarchical Reinforcement Learning

The objective here is to develop a more centralized controlling agent. In multi-agent systems, each agent independently provides its own solution at every time step. However, the change_bus action often overshadowed other actions, even in scenarios where it was not optimal.

We observed that, in some instances, the combined effects of multiple actions led to suboptimal outcomes compared to cases where either a single action or no action would have been more effective.

This improvement focuses on selectively determining when certain actions are unnecessary, allowing the system to refrain from taking action when appropriate. In particular states, the optimal choice is often to take no action. In such cases, the controlling agent assesses the actions proposed by individual agents and decides which, if any, should be executed based on its observations.

This adjustment also enhances the model's versatility, equipping it to provide a broader range of solutions during critical decision points.

For this enhancement, pretrained agents from the Multi-Agent System were utilized, as retraining them in the same way would have been redundant.

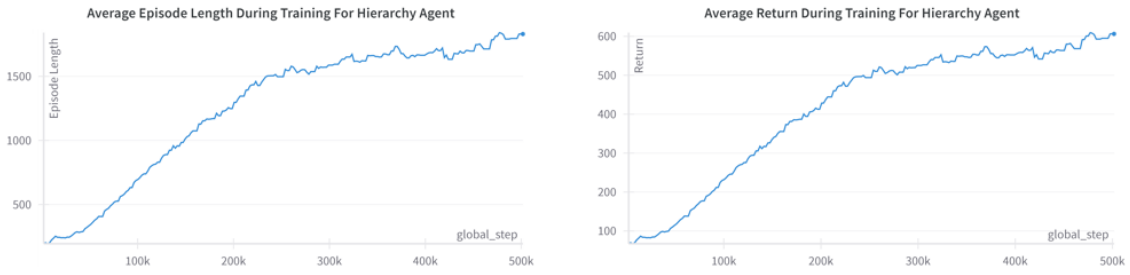

Figure 12: Average reward and episode length during training of Hierarchical agent

3.4 Improvement 4: Using GNN to extract features

As a final improvement, we noticed that GNN yielded promising results in A2C so we wanted to investigate the use of the GNN in the same way as done on the A2C implementation in reducing the observation space as well as encoding the graph structure into the observation space.

The goal of this improvement was to provide a better foundation for the multi-agent, and, by extension, the hierarchical agent through better features captured from the observations.

This is done for each of the Muti Agents and for the Hierarchical Agent which were also retrained using the updated observation space.

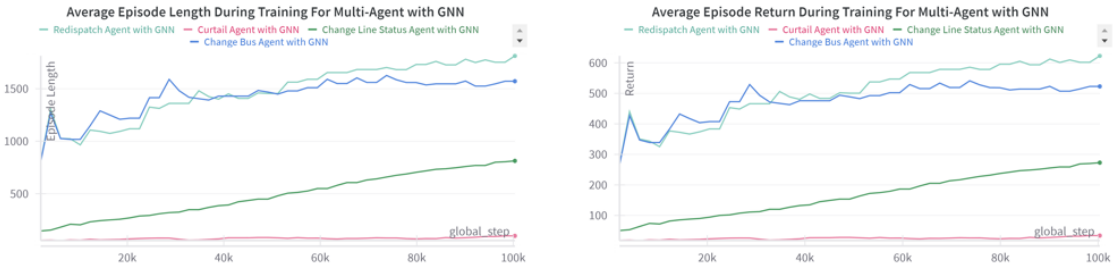

Figure 13: Average reward and episode length during training of Multi Agents with GNN

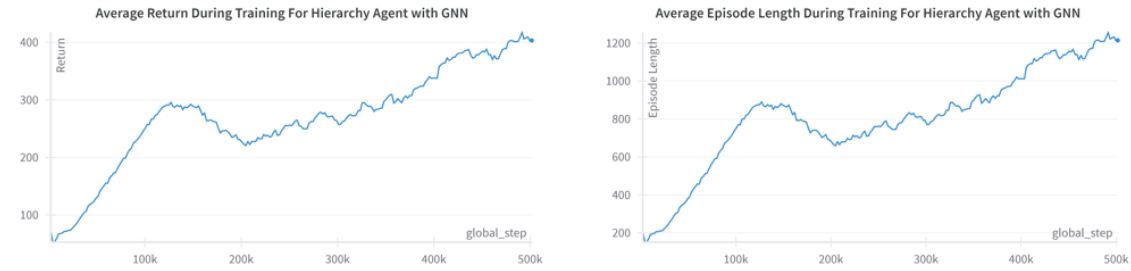

Figure 14: Average reward and episode length during training of Hierarchical Agent with GNN

Results for both were worse than the RL agents without the GNN. This may be due to the graph structure not adequately capturing the necessary latent space for PPO.

3.5 Comparisons

3.5.1 Training Times

Training times improved from the base agent to the first enhancement and further with the Multi-Agent approach, as agents could be trained in parallel with reduced action spaces. However, the Hierarchical Agent required longer training due to the added complexity of selecting which agent to deploy and when. The GNN increased training time for both Multi-Agent and Hierarchical agents and yielded poorer results than either method alone, though it still outperformed the Base Agent.

3.5.2 Performance

Despite modifications, no overall performance gains were observed, with a notable decline when combined with the GNN. Within the Multi-Agent system, the redispatch agent achieved better outcomes than other single-purpose agents.

3.5.3 Strengths and Weaknesses

Each improvement demonstrated strengths in reduced observation/action spaces and shorter training times compared to the base agent.

Strengths Multi-Agent and Hierarchical approaches can handle continuous action spaces effectively.

Weaknesses

- The first improvement introduced information loss due to discretized observations.
- The Multi-Agent System struggled with ineffective agent selection, leading to inappropriate actions. Independent training fostered a greedy policy, which can be suboptimal in a cooperative setting.
- The Hierarchical Agent's performance may have been hindered by isolated training of Multi-Agents, limiting synergy across agents.

4 Other

All implementations of A2C and PPO made use of stablebaselines3 with wrappers designed by ourselves.

5 Conclusion

This project explored two distinct approaches to power grid management through reinforcement learning: Actor-Critic (A2C) and Proximal Policy Optimization (PPO). Through various improvements and modifications to both architectures, we gained valuable insights into the challenges and potential solutions for managing complex power grid systems.

The A2C implementation demonstrated significant improvements through three key modifications. Normalization of the observation space and reduction of input features led to more efficient training, processing 168% more timesteps in 71% additional time compared to the baseline. The introduction of curriculum learning proved particularly effective, achieving stable training patterns and reaching consistent rewards around 544 while reducing training time by 77%. The incorporation of Graph Neural Networks showed promise in capturing the grid's structural information, though it exhibited some convergence inconsistency.

The PPO architecture, while theoretically well-suited for high-dimensional spaces and resource management, faced different challenges. The multi-agent system approach allowed for better handling of continuous action spaces and specialized behaviour, particularly in the redispatch agent. However, the hierarchical implementation, while addressing the coordination issues of independent agents, didn't yield the expected performance improvements. The integration of GNN with PPO actually decreased performance, suggesting that certain architectural combinations may not be universally beneficial.

A common thread across both implementations was the trade-off between complexity and performance. While reduced observation spaces and normalized inputs generally led to faster training times, they sometimes came at the cost of performance stability. The curriculum learning approach in A2C and the multi-agent system in PPO both demonstrated that breaking down complex problems into manageable components can be beneficial, though the effectiveness varies based on the specific architecture.

Future work could explore hybrid approaches that combine the stable learning characteristics of curriculum training with the specialized capabilities of multi-agent systems. Additionally, investigating alternative graph neural network architectures or different methods of incorporating structural information could potentially address the convergence issues observed in both implementations.

This study ultimately highlights the complexity of applying reinforcement learning to power grid management and demonstrates that different architectural approaches may be better suited for specific aspects of the problem. The success of curriculum learning in A2C and the specialized capabilities of the multi-agent PPO system suggest that a combination of these approaches might be the most promising direction for future development.

References

[BLCW09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[GSR+17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, 2017.