Reinforcement Learning Assignment Report

Luke Kerker 2460117 | Christine Bau 2463698 | Muhammad Sahal Goolam 2454262 | Rayhaan Hanslod 2430979

I. Introduction

Grid2Op is a framework designed to simulate the operation of real-world power grids, providing a platform for training intelligent agents to manage complex electrical networks [1]. In this project, we utilise Grid2Op to develop RL-based agents capable of handling key grid challenges, such as preventing outages, responding to equipment failures, and ensuring grid stability under dynamic conditions. This environment features an extremely large observation and action space, presenting significant challenges due to the curse of dimensionality for traditional reinforcement learning methods. In this work, we consider the employment of two primary policy gradient methods: Advantage Actor Critic (A2C) and Option Critic (OC) [2], considering various ablations of the observation and action spaces as well as in the algorithmic design. We publicly release our code at: https://github.com/Shnifel/RL-Assignment.

II. Environmental setup

A. Observation Space

The observation space in Grid2Op is exceptionally detailed, encompassing a wide range of attributes crucial for effective grid management. These attributes include static features, such as grid topology, network structure, and connectivity, as well as dynamic, real-time data on power flow, line loads, and bus voltages. This extensive information provides a comprehensive view of the grid's current state and operational conditions, supporting the agent's decision-making process. However, the complexity and volume of this data pose challenges in processing and interpreting the most relevant features.

B. Action Space

In this assignment, we analyzed and explored three distinct action spaces provided by Donnot in the Grid2Op environment to evaluate their suitability for grid-related operations [1]. The available action spaces—Discrete, Multi-Discrete, and Continuous—each offer unique methods for controlling various grid elements. Actions in Grid2Op range from switching operations, such as activating or deactivating specific lines or modifying bus configurations, to adjusting power flow distributions across the network. These actions allow the agent to interact with the environment at different levels of granularity, enabling both broad and highly specific modifications. By utilizing these diverse actions, agents can respond to grid conditions with varying levels of precision.

Table VI shows all the action features used for each action space.

C. Reward Structure

Our agents are evaluated using a combined, scaled reward that incorporates both the L2RPN reward and the N1 reward. The L2RPN reward is calculated as the sum of the squared margins for each powerline. The margin of a powerline is defined as the ratio of the difference between the thermal limit and the flow to the thermal limit, provided that the flow is within the thermal limit. If the flow exceeds the thermal limit, the margin is set to zero. The reward is then the sum of the squared margins across all powerlines.

The N1 reward measures the maximum flow across all powerlines after a specified powerline has been disconnected.

III. Advantage Actor Critic (A2C)

Actor-Critic is a reinforcement learning algorithm that combines the strengths of both value-based and policy-based methods. It uses two separate components: the actor, which determines the best action to take in a given state by learning a policy, and the critic, which evaluates the quality of actions by estimating the value function for the current policy. The actor updates its policy based on feedback from the critic, allowing it to gradually improve its actions, while the critic refines its value estimates based on rewards received. The parameters of the actor and critic are updated at the same time. This combination enables actor-critic to achieve stable learning and effective policy improvement. The model's hyperparameters are listed in VII.

The making and improvement of this agent was carried out in the following stages:

1) **Baselines (III-A):** Initial benchmarks identify the most effective action space and set a baseline for adjustments.
2) **Reduced Observation Space (III-B):** Dimensionality reduction is assessed to remove non-essential features without degrading performance.
3) **Action Visualization (III-C):** Visualization of actions highlights features that impact episode longevity, guiding action refinement.
4) **Reduced Action Space (III-D):** Key action features in the Multi-Discrete space are isolated to enhance agent efficiency.
5) **Multi-Agent Setup (III-E):** Specialized agents coordinate distinct tasks, improving grid control.
6) **Limitations and Issues (III-F):** Challenges in performance consistency and computational limitations are discussed.

A. Baselines

Figures 1 and 2 illustrate the rewards and episode lengths obtained during training with all action and observation features, both of which have been smoothed for clarity. A directly proportional relationship is evident between episode length and the associated rewards, with the Multi-Discrete action space demonstrating the fastest convergence among all action spaces. Additionally, the Discrete action space achieves the highest average reward and episode length.

Table I demonstrates that the Multi-Discrete action space achieves the highest episode rewards and lengths among all action spaces. We chose to concentrate our further adjustments on the Multi-Discrete action space to utilize its faster convergence and its ability to achieve higher episode rewards and longer episode lengths.

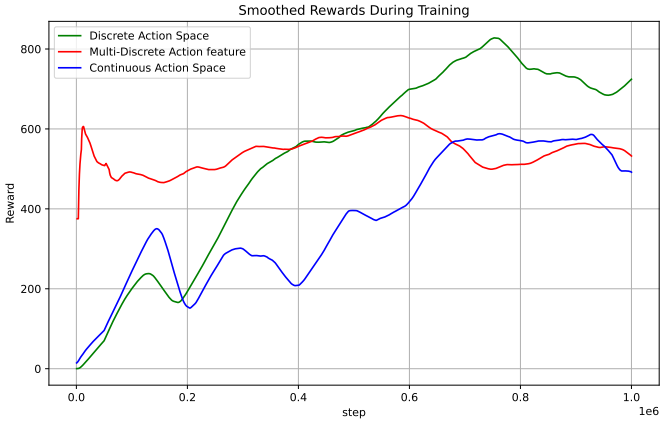

Fig. 1. Episode Rewards of different Action spaces on entire observation space

Fig. 2. Episode Lengths of different Action spaces on entire observation space

B. Reduced Observation Space

Through our analysis of the L2RPN and N1 rewards, we identified observation features that are more or less likely to influence agent performance. Specifically, we determined that features related to alarms, alerts, storage, voltage angles, and maintenance are unlikely to significantly affect agent performance, as they do not directly, or even indirectly, provide meaningful information about the powerline flow or margin. Hence we removed the observation features listed in section A.

TABLE I
COMPARISON BETWEEN MAX ACTION SPACE REWARDS AND LENGTHS PER EPISODE DURING TRAINING

Action Space	Max Reward	Max Length
Discrete	851	2457
Multi-Discrete	1008	3064
Continuous	620	1794

Fig. 3. Episode Rewards of different Observation spaces on entire observation space

Fig. 4. Episode Lengths of different Observation spaces features on entire observation space

TABLE II
COMPARISON BETWEEN MAX REWARDS AND LENGTHS PER EPISODE DURING TRAINING OF AGENTS WITH DIFFERENT OBSERVATION SPACES

Observation Space	Max Reward	Max Length
Entire Observation Space	645	1951
Reduced Observation Space	642	1939

Figures 3 and 4 present the episode rewards and lengths of agents trained with different observation spaces. The results show minimal variation between the performance of the two agents, with the agent using the reduced observation space performing only slightly worse. Table II supports this finding, as the difference in maximum rewards and episode lengths between the agents is negligible. Consequently, based on our findings, we decided to utilize the entire observation space and focus on adjusting and improving the action space. Any potential gains from modifying the observation space would require a time-consuming process of eliminating individual observation features, which is neither computationally nor time efficient.

C. Action Visualization

To deepen our understanding of the Grid2Op environment, we aimed to visualize both effective and ineffective actions within the continuous action space. Specifically, we plotted actions that led to episode termination alongside all actions taken in episodes that exceeded one thousand steps (excluding the final action). We selected the continuous action space for this experiment due to its relatively low dimensionality compared to the multi-discrete and discrete action space, which allowed us to reduce the action space to six action features. This reduction was achieved by excluding features related to alarms, alerts, and storage, as they do not directly impact powerline flow or margin.

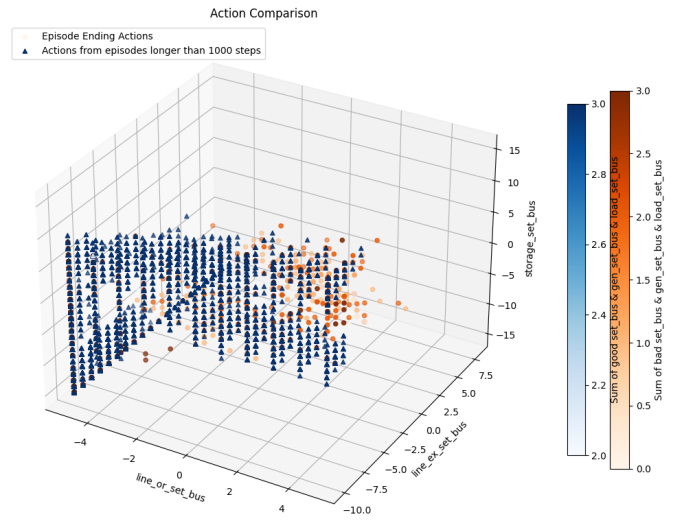

Fig. 5. Action Comparison between episode ending actions and actions taken during long episodes

Figure 5 compares effective and ineffective actions. Notably, in effective actions, the features ”set_bus,” ”gen_set_bus,” and ”load_set_bus” (whose values all range from 0 to 1) are never set to zero, as indicated by the blue color range sum spanning from 2 to 3. In contrast, the sum of these features vary between 0 and 3 for ineffective actions. Despite this difference, no clear pattern emerges from the overall distribution of the two action types, as they are generally similar in most cases. This finding supports our hypothesis that the utility of most actions is largely dependent on the specific state context.

This visualization provides insightful comparisons between effective and ineffective actions. The features ”set_bus”, ”gen_set_bus,” and ”load_set_bus” appear to play critical roles in sustaining long episodes. For effective actions, the fact that these actions are consistently non-zero suggests that they are necessary for stabilizing grid operations. These findings aids in decisions when we are developing the agents and setting action constraints that prioritizes bus-related features that can enhance the episode longevity.

D. Reduced Action Space

We conducted an in-depth analysis of the L2RPN and N1 rewards to derive insights into the key action features influencing agent performance within the Multi-Discrete action space. Through extensive experimentation, we identified the action features ”set_line_status,” ”set_bus,” and ”sub_set_bus” as the most influential in affecting agent performance. Additionally, we observed that the ”set” and ”change” action features can be used interchangeably with comparable performance. This finding confirmed our understanding of these action features, as the key distinction between these features is that the ”set” actions allow for specifying the exact change of connection, while the ”change” actions are limited to simple connect or disconnect actions.

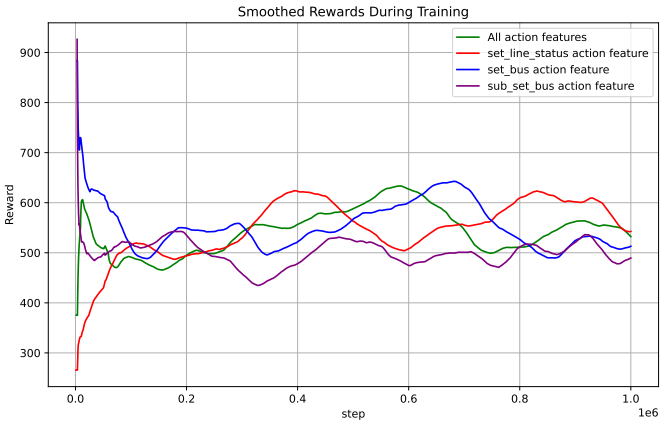

Fig. 6. Episode Rewards of different Multi-Discrete Action features on entire observation space

TABLE III
COMPARISON BETWEEN DIFFERENT MULTI-DISCRETE ACTION FEATURES

Action Space	Max Reward	Max Length
All action features	1008	3064
Set line status action feature	645	1951
set bus action feature	883	2689
sub set bus action feature	926	2821

Figures 6 and 7 depict the rewards and episode lengths during the training of agents with various action features. Notably, agents utilizing individual action features achieved performance comparable to the agent employing all action

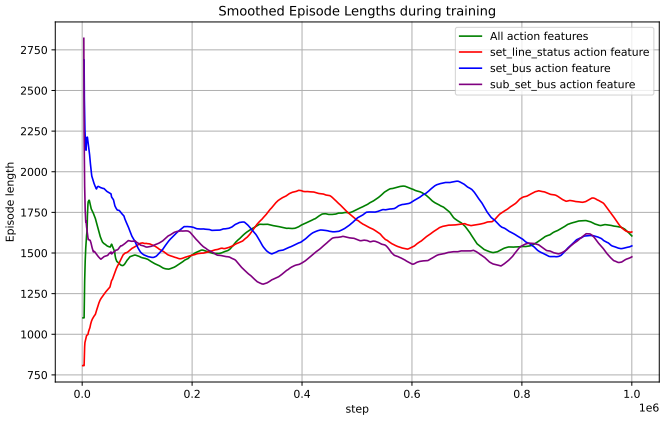

Fig. 7. Episode Lengths of different Multi-Discrete Action features on entire observation space

features. This finding suggests that the actor-critic model is not fully leveraging the potential of each action feature. A likely contributing factor is the high dimensionality of the action space when using all action features and the large observation space, which causes the development of sub-optimal policies within the agent.

Table III presents the maximum episode rewards and lengths for agents utilizing different action features during training. While none of the single action feature agents outperformed the agent using all action features, the difference between the "sub_set_bus" agent and the full action feature agent was minimal. Moreover, all single action feature agents demonstrated strong performance. These findings further indicate that the actor-critic model is not fully exploiting the capabilities of each action feature to its maximum potential.

E. Multi-Agent

Multi-agent systems consist of many decision-making agents that can interact within a shared environment to achieve common goals. They can be used for task specialisation and parallel processing. For our task, we created three separate environments for agents pretrained using A2C, each responsible for a different task:

1) **Set Line Status Agent:** This agent operates within the `set_line_status` action feature and is responsible for controlling actions that set the power line status.
2) **Set Bus Agent:** This agent operates within the `set_bus` action feature. This allows it to reconfigure buses in substations.
3) **Set Substation Bus Agent:** This agent controls substation bus assignments within the `sub_set_bus` action feature and reconfigures substations dynamically.

For each episode, the agents receive observations from the Grid2Op environment and the three separate models predict their respective actions. These actions are then concatenated into a single action and executed. This approach emulates curriculum learning by allowing each agent to specialise in a specific task while working in coordination with the other agents. Our motivation for adopting a multi-agent approach stems from our exploration of the reduced action space, where we observed that individual action features were not being fully utilized to their maximum potential. Additionally, the successful application of multi-agent methods by top-performing agents in the NeurIPS 2020 competition further encouraged us to implement this strategy [3].

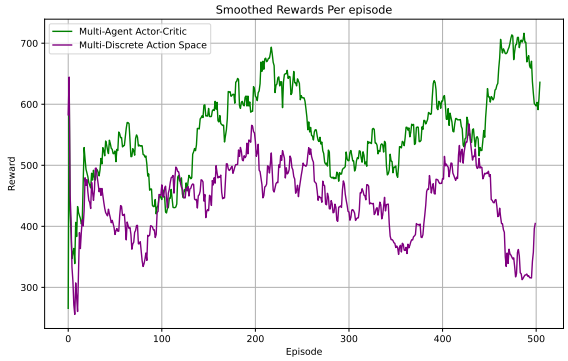

Fig. 8. Episode Rewards of Multi-Agent Actor-Critic and normal Actor-Critic

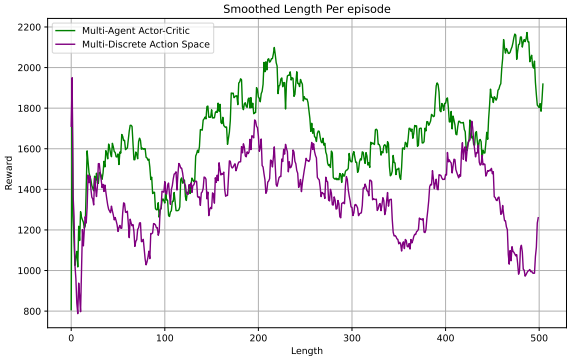

Fig. 9. Episode Lengths of of Multi-Agent Actor-Critic and normal Actor-Critic

TABLE IV
COMPARISON BETWEEN MULTI-AGENT AND BASELINE

Agent	Max Reward	Ave Reward	Max Length	Ave length
Multi-Agent	2638	561	8064	1695
Multi-Discrete Baseline	2249	447	6905	1379

Figures 8 and 9 compare the evaluation results of the multi-agent and the full observation, Multi-Discrete action space agent. As shown in both the figures and Table IV, the Multi-Agent significantly outperforms the Multi-Discrete action space agent. The Multi-Agent achieves higher maximum rewards and episode lengths, as well as greater average rewards and lengths. Notably, it sustains operations in the Grid2Op environment for twenty-eight consecutive days, exceeding the base agent's performance by four days.

F. Limitation and issues

Maintaining consistent performance across all models proved challenging, as both training and evaluation exhibited multiple instances of episode lengths and rewards approaching zero, which significantly impacted the models' overall performance. Due to computational constraints and time limitations, we were unable to extend model training beyond one million steps. For example, training the Multi-Discrete Action Space model with full observation took approximately eighteen hours to complete one million steps. These constraints restricted our ability to apply methods and techniques that necessitate more extensive computational resources and prolonged convergence times.

Despite the superior performance of the multi-agent system, we recognize that concatenating the actions of each agent may have limitations and implications, leading to the possibility that a more refined learning approach could yield better results.

IV. Option Critic

Options [4] in RL refers to a framework allowing agents to take higher-level, temporally extended actions known as options, as opposed to primitive actions. This allows for the execution of sequences of actions as single cohesive units and for the transfer of skills. While this is an attractive framework, concisely defining "good" options remains a challenging task, as in the case of this environment where we might for instance want to be able to execute a power redistribution in the context of a blackout. The Option-Critic [2] architecture extends the options framework, allowing for the automation and learning of useful options and termination conditions for each option. This model, evaluates and improves option policies and termination conditions, similar to how actor-critic methods evaluate and improve actions using a critic. As such, we utilise this framework so as to impose hierarchical structure over our agent, allowing for the learning of high-level decisions, as well as how to act within an option, and hence enhancing the adaptibility in our decisions.

The investigation and improvement of the Option-Critic model proceeds through these stages:

1) **Architecture and Training (IV-A):** Overview of the Option-Critic network, including option policies and training for selection and termination.
2) **Model Baseline (IV-B):** Compare discrete and continuous action agents to assess rewards and focus on continuous actions.
3) **Observation Culling (IV-C):** Optimize performance by evaluating and reducing observation spaces.
4) **Attention Incorporation (IV-D):** Investigate attention mechanisms to enhance focus on key observations.
5) **Additional Considerations (IV-E):** Adjust Q-network architecture and action reparameterization for better adaptability and rewards.

A. Architecture and Training

Training

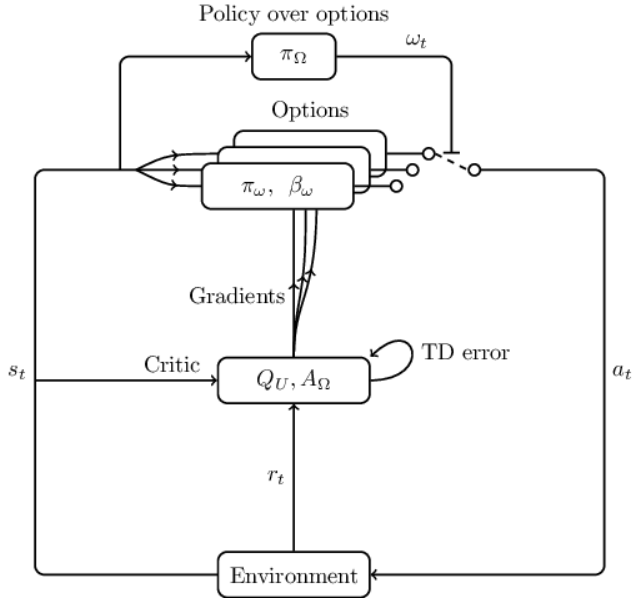

Fig. 10. Option-Critic Training Mechanism [2]

Figure 10 illustrates the architecture of the option-critic network, and is composed of two main components:

- **Options Actor** - The options actor is responsible for learning a policy over the set of options Ω, and intra-option policies and termination conditions for each $\omega \in \Omega$ that seeks to maximise the return at both the macro level of selecting valuable options and the micro level of the primitive sequence of actions within each option.
- **Critic** - The critic evaluates both the options and intra-option policies of the actor. The critic estimates a Q-value function, Q_U, over state-action pairs for each option, (s, a, ω), as well as an advantage function measuring the value of choosing a particular option in a given state, A_Ω.

The Option-Critic algorithm with intra-option Q-learning begins by initialising parameters for the options, intra-option policies and termination conditions, selecting an option ω based on an ϵ-greedy policy over options. At each step, the agent chooses an action a according to the intra-option policy for the current option $\pi_\omega(a|s)$, executes a, and observes a new state s' and reward r.

The option evaluation phase (critic) then calculates a temporal-difference error δ to update the Q-value of the state-option pair $Q_U(s, \omega, a)$. If the new state is non-terminal, δ is adjusted with terms that account for continuing the current option and for switching to the best alternative.

The option improvement phase (options actor) then updates the parameters for the intra-option policy to favour actions that improve the Q-value, and adjusts the termination policy by minimising the difference between the option value and the overall value function. If the option terminates in the new state, a new option is selected, and the process repeats

until a terminal state is reached. This iterative evaluation and improvement enable the agent to learn hierarchical policies that effectively solve complex tasks.

Implementation Details

We utilise an existing implementation of the option critic for training[1].

Actor architecture - The actor is composed of a 2 fully-connected layer with $ReLU$ and $tanh$ activation functions respectively. This acts as a feature extractor and is common across all the options. Following this, the network computes distinctly for the current running option ω an output $W_\omega \mathbf{x} + \mathbf{b}_\omega$, where $\mathbf{x}$ is the output of the feature extraction. In the discrete action output case, W_ω and $\mathbf{b}_\omega$ are interpreted in the context of a logistic regression, whereas in the continuous action case which we implement, W_ω is used to estimate the mean of a Gaussian distribution and b_ω for the standard deviation.

Option critic architecture - The state-option pairs (s, ω) is implemented as a single layer mapping features to Q-values for each state-option.

While such an algorithm has significant potential for learning hierarchical structure, there are several issues inherent in the training of this network:

- Network instability - The training and updating of two networks with a feedback loop simultaneously poses significant challenges to network stability. Network freezing of the Q-value network is employed to mitigate the effect of this.
- Options termination - While the usage of a soft option selection policy allows for exploration of other options, there remains no guarantees on the termination of an option and hence it is possible that a particular option could continue indefinitely. As such, a maximum option length is set initially on each option as the network is in the earlier phases of learning.
- Number of options - The specification of the number of options is no trivial task as we depend on heuristics and have no theoretical guarantees on what kind of options will materialise in the agent. For the purposes of this study, owing largely to compute and the rise in computational demand with larger options, we restrict the number of options to 2 options. However, we do in certain instances contrast the effect of additional options.

Additional extensions are reserved for discussion with the their corresponding results. A comprehensive list of hyperparameters is presented in Table VIII.

[1]The implementation can be found at : https://github.com/lweitkamp/option-critic-pytorch

B. Model baseline

In order to obtain a baseline comparison, we trained two agents that execute discrete and continuous actions in the environment. With respect to the discrete action space, we restrict ourselves to the subset of actions that were found to be useful, i.e. `set_line_status` and `set_line_bus`. For the continuous actions, we utilise the `redispatch` and `curtail` actions within our action space which are responsible for modulating the power supply for each generator. The reward training curves are illustrated in Figure 11.
Figure 11 demonstrates that the agent operating within the discrete action space performed significantly worse than its counterpart in the continuous action space. However, the discrete action space agent exhibited greater stability, as evidenced by its considerably lower reward variance. We chose to concentrate our adjustments on the continuous action space to leverage its superior reward capacity. Additionally, we aim to explore how the continuous action space can enhance the performance of continuous action features.

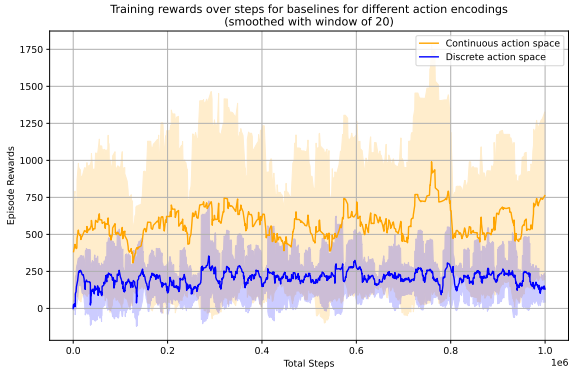

Fig. 11. Baseline training reward curves of option-critic (1 standard deviation shading is shown)

C. Observation culling

We sought to reduce the observation space, similar to the approach used in the Actor-Critic method, by removing the observation features identified in list A. In addition, we eliminated other action features that do not directly impact powerline flow or margin, such as temporal features, to maintain the Markov property, and network topology features, which are large and sparse due to their representation of the network's structure. However, we acknowledge that this significant reduction in information may impact our agent's performance and generalization capabilities.

Figure 12 illustrates the rewards during training for both the full observation agent and the culled observation agent. Notably, the performance difference between the two agents is minimal, suggesting that many features in the observation space do not contribute significantly to understanding powerline flow and margin.

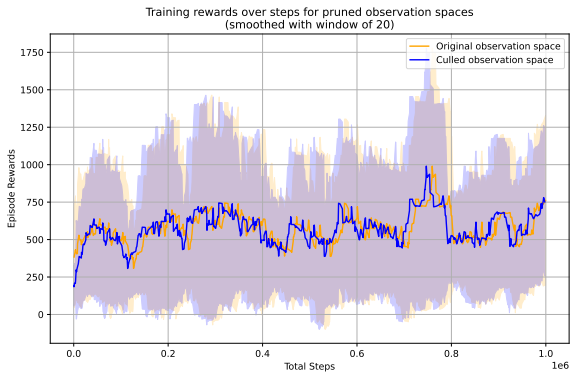

Fig. 12. Training rewards of original observation space and culled observation space

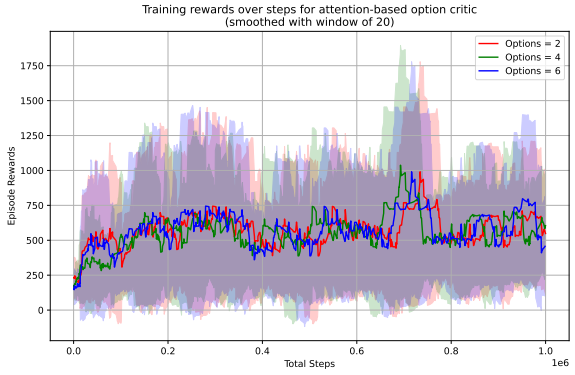

Fig. 13. Training rewards of attention incorporation in actor critic (1 standard deviation shaded)

D. Attention incorporation

Considering the marginal performance gains exhibited above, we further explored the incorporation of more fine-tuned and relevant observation features that would allow for increased performance. We implemented a gating mechanism, inspired from Attention Option-Critic [5], which is unique for each option. Through this, we aim to only attenuate relevant observations, but additionally make this option tailored. This is accomplished through the introduction of an additional option specific fully-connected layer with a sigmoid activation following the feature extraction which produces a vector of weightings with each value $\in [0, 1]$. This is then multiplied by the feature vector.

Figure 13 exhibits the training rewards under the attention mechanism. We additionally contrast this under differing numbers of options. As exhibited, we observe performance in line with that of the previous step, with the peak performances remaining similar. This failure of the attention mechanism can be attributed to several factors. Firstly, the lack of stability inherent in training is further exacerbated by the introduction of additional learnable parameters. However, we note that the model maintained comparable performance despite this introduction. Secondly, the performance limitations may stem from over-parameterisation when adding option-specific gating layers, leading to potential overfitting to irrelevant features and less generalisable option policies. While the gating mechanism allows each option to focus on unique aspects of the observation space, it also increases the learning complexity, making convergence more challenging. This suggests that while adding attention mechanisms can improve feature selectivity, they may need additional regularization techniques to promote diversity. Finally, it's possible that the interaction between options is hindered by a lack of coordination across the gating mechanisms, where each option independently attenuates features without a shared understanding of the overall task. This could limit the emergence of complementary skills across options.

With respect to the varying number of options, we again observe marginal differences, which we attribute to similar reasons as above. This implies a lack of diversity in learnt attentions. Additionally, an increased number of options would automatically require longer training in order to elicit more nuanced options.

E. Additional considerations

We explored additional adjustments in light of the issues faced above, with adjustments yielded marginal or no improvements. We modified the architecture of the Q-Network by increasing the number of linear layers to two and incorporating dropout regularisation within the context of the attention adjustment. The insight behind this was to augment the model with more capacity as the relatively shallow policy networks may not learn meaningful associations. Figure 14 presents the rewards achieved by the modified Q-network architecture agent. It is evident that the performance of the Q-network agent is only marginally different from that of the baseline.

Reward shaping was implemented by incorporating the episode lengths as a reward at the end of each episode, which incentivises the agent based on the duration of the episode. This follows the nature of the problem that longer episodes result in greater rewards, with the lack of such being encouraged in the existing reward function. We anticipated that this reward structure would enhance our agent's performance, given this directly proportionality. Figure 15 illustrates the rewards attained by the reward-shaped agent. It is clear that the performance of the reward-shaped agent further exhibits minimal differences.

Lastly, we considered a reparameterisation of the action space. This followed the observation that some actions taken in regards to changing curtailment were invalid, as a result of sampling our actions from a Gaussian distribution. To curb this, we reparameterised our actions as scalars in the range $[0, 1]$, and mapped this to the range of each action respectively. Figure 16 exhibits this. We highlight marginally improved performance, however it remains insufficiently different to the baseline.

Fig. 14. Training rewards of Modified Q-Network architecture

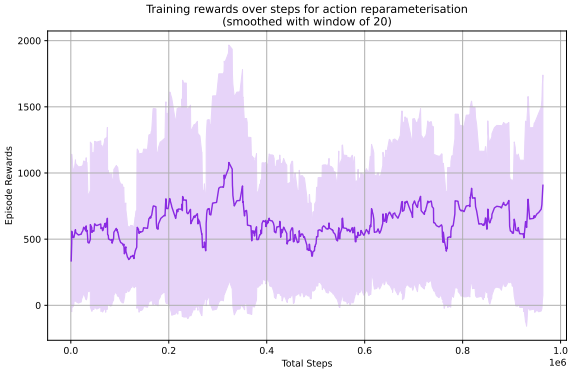

Fig. 16. Training rewards of action reparametrization agent

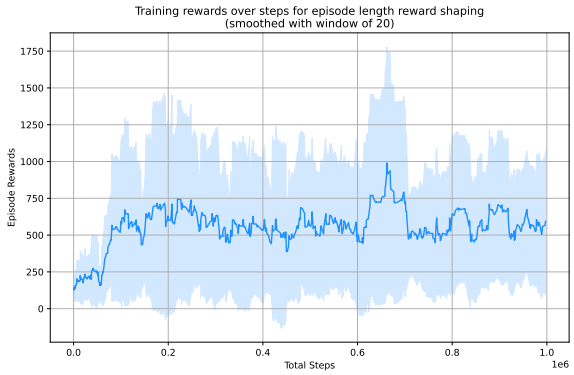

Fig. 15. Training results of reward shaping

F. Discussion and limitations

TABLE V
MEAN REWARD WITH STANDARD DEVIATION FOR LAST 100 EPISODES OF EACH AGENT

Model variant	**Mean Reward ± Std Dev**
Baseline (Continuous)	576.94 ± 353.78
Baseline (Discrete)	151.29 ± 97.02
Pruned observation space	523.70 ± 386.20
Attention (2 options)	474.05 ± 385.98
Attention (4 options)	593.25 ± 668.46
Attention (6 options)	442.17 ± 219.24
Modified Q-Network	531.03 ± 370.45
Action reparameterisation	848.17 ± 640.38
Reward shaping	673.57 ± 687.14

Our exploration of the Option-Critic (OC) architecture within this framework has highlighted a range of challenges and potential areas for improvement, especially in the context of complex environments like power grid management. Table V depicts a summative evaluation of the modifications explored within this framework of the last 100 training episodes. We highlight the higher gains of the action reparameterisation, traded off for a much higher variance relative to the the others. This work illustrates that while Option-Critic presents an intriguing framework for hierarchical reinforcement learning, several factors hinder its effectiveness, particularly when operating across discrete and continuous action spaces and with diverse observational requirements.

- **Challenges in Continuous and Discrete Observation Spaces**: One of the core challenges encountered was handling the vast and detailed observation space provided by Grid2Op. Continuous observations, while more informative, add substantial computational load and increase the risk of overfitting as the agent tries to extract meaningful patterns from high-dimensional, dense data. Conversely, discrete observations simplify learning but may omit critical subtleties required for effective decision-making, resulting in less adaptable policies. Reducing the observation space was partially successful; however, the removal of certain features, even if marginally relevant, occasionally led to a noticeable drop in agent performance. This underlines the necessity of retaining a delicate balance between complexity and sufficiency of observational features.
- **Difficulties in Learning Optimal Options:** Learning useful options in the Option-Critic architecture remains inherently challenging. The option-learning mechanism must navigate between high-level actions and granular within-option policies, which often leads to instability, especially as the number of options increases. Introducing more options theoretically increases flexibility, but, as observed, also increases the learning difficulty due to the need to train each option's intra-policy and termination conditions. The results suggest that an optimal balance between flexibility and stability is yet to be achieved.
- **Lack of Interpretability in Learned Options:** One of the limitations we observed was the lack of interpretability in the learned options. With the attention mechanism, the agent was able to focus on certain aspects of the observation space per option, but the actual decision process within each option remained opaque. This made it difficult to understand the specific contribution of each option toward the overall policy and posed a challenge for debugging and fine-tuning. Additionally, the gating

mechanism introduced with attention did not yield substantial performance improvements, which could be due to overfitting on less relevant features. This highlights the need for future work to incorporate methods that both interpret and regularize option policies for improved transparency and generalization.

Despite the lack of performance gains, the experiments with Option-Critic and attention mechanisms yielded valuable insights. Our experiments suggest multiple areas for future work. Additional regularisation techniques (e.g., dropout, weight decay) may prevent overfitting in attention-gated mechanisms, while parameter sharing across options could reduce complexity and enhance training stability. Furthermore, coordination across options, possibly through hierarchical or joint training of gating mechanisms, may encourage complementary option policies, leading to a more coherent and efficient policy overall. Furthermore, the incorporation of such an agent within the multi-agent framework observed in the previous section may boost the relatively high performance even more, as this agent can execute the continuous actions lacking on that front.

V. Conclusion

In this work, we investigated two distinct reinforcement learning paradigms for power grid management within the Grid2Op framework: the Advantage Actor-Critic (A2C) and Option-Critic architectures. These approaches offered different perspectives on handling the complexities of power grid control. The A2C implementation proved effective through its multi-agent design. We achieved some performance gain by strategically decomposing the control problem into specialized agents, each managing specific aspects of grid operations. This distributed approach significantly outperformed traditional single-agent implementations, demonstrating the value of targeted expertise in complex control scenarios.

While the Option-Critic architecture presented an innovative approach to temporal abstraction in grid management, it faced unique challenges in balancing hierarchical control with practical implementation. Through extensive experimentation with observation spaces, attention mechanisms, and architectural variations, we gained valuable insights into the framework's capabilities and limitations in real-world applications.

Our findings underscore both the remarkable potential of reinforcement learning in power grid management and the critical importance of architectural decisions in practical implementations. The improvements of our multi-agent approach, in particular, highlights how thoughtful system design can effectively address the challenges of complex real-world control problems.

References

[1] B. Donnot, "Grid2op-a testbed platform to model sequential decision making in power systems," *GitHub repository*, 2020.

[2] P.-L. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[3] A. Marot, B. Donnot, G. Dulac-Arnold, A. Kelly, A. O'Sullivan, J. Viebahn, M. Awad, I. Guyon, P. Panciatici, and C. Romero, "Learning to run a power network challenge: a retrospective analysis," in *NeurIPS 2020 Competition and Demonstration Track*. PMLR, 2021, pp. 112–132.

[4] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[5] R. Chunduru and D. Precup, "Attention option-critic," *arXiv preprint arXiv:2201.02628*, 2022.

Appendix

Action Feature	Discrete	Multi-Discrete	Continuous
change_bus	✓	✓	✓
change_line_status	✓	✓	✓
curtail	✓		✓
curtail_mw	✓		✓
redispatch	✓		✓
set_bus	✓	✓	✓
set_line_status	✓	✓	✓
set_line_status_simple	✓		
set_storage	✓		✓
one_line_change		✓	
one_line_set		✓	
one_sub_change		✓	
one_sub_set		✓	
raise_alarm		✓	✓
raise_alert		✓	✓
sub_change_bus		✓	
sub_set_bus		✓	

TABLE VI
Action Features in Discrete, Multi-Discrete and Continuous Spaces

Observation features removed in Actor Critic

- was_alarm_used_after_game_over
- was_alert_used_after_attack
- time_before_cooldown_line
- time_before_cooldown_sub
- time_next_maintenance
- time_since_last_alarm
- time_since_last_alert
- time_since_last_attack
- timestep_overflow
- storage_charge
- storage_power
- storage_power_target
- storage_theta
- duration_next_maintenance
- gen_theta
- load_theta
- theta_or
- theta_ex
- total_number_of_alert

Parameter	Default Value	Description
Policy	MlpPolicy	Neural network policy type
Learning Rate	7e-4	Step size for optimizer
momentum	0	Accelerates convergence by smoothing updates
n_steps	5	Number of steps per update
gamma	0.99	Discount factor
ent_coef	0.01	Entropy coefficient for exploration
vf_coef	0.25	Value function coefficient
max_grad_norm	0.5	Maximum gradient norm for clipping
rms_prop_eps	1e-5	RMSprop optimizer epsilon
use_rms_prop	True	Use RMSprop optimizer
use_sde	False	State Dependent Exploration
sde_sample_freq	-1	Sample frequency for SDE
normalize_advantage	False	Whether to normalize advantage estimates
seed	0	Controls randomization for reproducibility

TABLE VII
HYPERPARAMETERS USED IN STABLE BASELINES 3 A2C IMPLEMENTATION

Parameter	Value	Description
Learning Rate	0.0005	Optimizer learning rate
Gamma	0.99	Discount factor
Number of Options	2	Number of options available to the agent
Temperature	1.0	Softmax temperature parameter for action distribution
Batch Size	32	Size of training batches
Replay Buffer Size	10,000	Maximum number of steps stored in replay buffer
Update Frequency	4	Number of actions before each SGD update
Freeze Interval	200	Interval between target network updates
Termination Reg.	0.01	Regularization to decrease termination probability
Entropy Reg.	0.01	Regularization to increase policy entropy
ϵ Start	1.0	Initial exploration rate
ϵ Min	0.1	Minimum exploration rate
ϵ Decay	20,000	Number of steps to decay epsilon to minimum
Optimal ϵ	0.05	Epsilon value when playing optimally
Optimizer	RMSprop	Optimization algorithm
seed	0	Controls randomization for reproducibility

TABLE VIII
HYPERPARAMETERS USED IN THE OPTION-CRITIC IMPLEMENTATION