COMPUTER SCIENCE DEPARTMENT

Reinforcement Learning

MiniGrid Final Project

Authors:
Tamir Houri
Almog Zemach

Feb. 2025

Contents

1 Algorithm Selection and Explanation

1.1 Algorithm Overview

Proximal Policy Optimization (PPO) PPO is a policy-gradient method that optimizes a surrogate objective to improve training stability. It uses clipped probability ratios to prevent large policy updates, ensuring more stable and sample-efficient learning:

$$L_t^{CLIP}(\theta) = \hat{E}_t \left[min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{1.1}$$

Where $\hat{A}_t$ is the Advantage and $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}}$. PPO's on-policy nature and robust performance make it suitable for the MiniGrid environment, where partial observability requires efficient exploration. See PPO Agent in notebook for implementation.

Dueling Deep Q-Network (Dueling DQN) Dueling DQN introduces a separate estimation of state-value and action-advantages, enhancing performance in environments with redundant state information:

$$\mathcal{L}(\theta) = \mathbb{E}\left[(y - Q(s,a;\theta))^2\right] \tag{1.2}$$

$$y = r + \gamma \max_{a'} Q_{\text{target}}(s', a') \tag{1.3}$$

Where $y = r$ if s' is terminal. The algorithm's off-policy nature and experience replay capabilities help address the sparse reward challenges of the MiniGrid environment. See Dueling DQN Agent in notebook for implementation.

1.2 Hyperparameters

Key hyperparameters were selected through empirical testing:

- **PPO:** learning rate = 2.5×10^{-4}, $\gamma = 0.99$, Clip Ratio = 0.2.
- **Dueling DQN:** learning rate = 1×10^{-4}, $\gamma = 0.99$, epsilon decay = 0.9999, replay buffer size = 20,000, target update frequency = 10 steps.

Both agents had mini-batch size of 64. The PPO clip ratio of 0.2 stabilized learning by preventing policy oscillations and gave us faster learning than 0.1 and 0.3. The Dueling DQN's epsilon decay of 0.9999 facilitated effective exploration, but fast enough to achieve more exploitation in advance epochs. We tuned the learning rate to optimize the learning process and to prevent vanishing/exploding gradient.

1.3 Deep Learning Architecture

Both models share a **three-layer convolutional feature extractor** (See Notebook) 16, 32, and 64 filters, all ReLU-activated, followed by a **Flatten** layer. This design captures essential spatial information from the $56 \times 56 \times 3$ tensor while keeping the parameter count manageable and effectively capturing key local features.

- **PPO**: Two separate networks (actor and critic) each use the same convolutional base but add a **64-unit Dense** layer (ReLU). The Actor outputs a softmax distribution over actions, while the Critic outputs a single scalar value estimate (See Appendix. A.1).
- **Dueling DQN**: A **128-unit Dense** layer (ReLU + LayerNorm) feeds two heads: Value $V(s)$ which outputs a single scalar and Advantage $A(s, a)$ which outputs a vector, with one entry per action. We combine them to form Q-values (See Appendix. A.2).

Validation & Experimentation

- Monitored average **rewards**, **losses**, and **episode lengths** per epoch throughout the training with visualizations.
- Compared different kernel sizes 5×5 and 7×7 along with filter counts. Smaller kernels converged more quickly on the $56 \times 56 \times 3$ tensor.
- Tested different layer sizes and different architecture including only fully connected layers which does not perform well.

1.4 Comparison of Algorithms

We investigated both **PPO** and **Dueling DQN** across progressively larger MiniGrid environments (small $\rightarrow$ medium $\rightarrow$ large). While each algorithm ultimately learned a successful policy in the last setting, they exhibited distinct behaviors and tuning requirements:

- **PPO**: Initially learned more slowly, but did not require extensive hyperparameter tuning or additional incentives to converge. By gradually increasing environment complexity (from small to medium to large), PPO consistently improved and achieved outstanding final results of 91% accuracy (See Appendix. A.3).
- **Dueling DQN**: Converged faster on smaller environments but struggled to transition to medium-sized environments. Despite significant hyperparameter tuning, it often failed to learn without adding reward shaping. suggesting higher sensitivity to environment scale and sparse reward distribution (See Appendix. A.3).

2 Performance and Analysis

Several key factors influenced the agents performance:

- PPO employs a stochastic policy, which enhances exploration—a crucial factor in partially observable environments with sparse rewards.
- Dueling DQN leverages a replay buffer, allowing it to train on a diverse set of room setups, resulting in more stable learning. However, Dueling DQN's ϵ-greedy policy was not effective in sparse reward environments, requiring us to adjust the reward structure to facilitate learning.

We adjusted training epochs and episodes per room size based on learning progress, typically using 50 episodes per epoch, yielding key insights:

- Monitoring the loss function provided valuable insights. A consistently increasing or excessively large loss often correlated with poor performance, prompting us to terminate such runs early.
- Initially, we used a fixed environment seed to allow the agents to focus on a single setup per epoch. However, this led to poor generalization, as the agents became overly specialized to specific room configurations.
- Initially, we saved the model weights corresponding to the best average reward per epoch. However, these weights often performed worse than the final training epoch's weights. This suggested that the learning process captures additional nuances in later epochs that the reward metric alone does not fully reflect.

3 Visualization and Graphs

- **Figure A.5: Average Reward vs. Epoch.** Notably, Dueling DQN demonstrates faster convergence than PPO across all environments.
- **Figure A.6: Average Steps vs. Epoch.** The average number of steps per episode decreases more rapidly for Dueling DQN than for PPO, indicating a faster learning rate and improved policy efficiency.
- **Figure A.7: Loss vs. Epoch.** While the loss metric exhibits significant variance, it provides limited insight into model performance, except in cases of exploding gradients, where a sudden spike in loss values can be observed.
- **Figure A.8: ϵ vs. Epoch.** The ϵ parameter influences the exploration phase of the Dueling DQN model, particularly during the early training stages when the model has not yet learned an effective policy. The sharp initial increase in the average reward (see Fig. A.5) aligns with the high values of the ϵ-greedy policy, which encourages exploration before gradually shifting toward exploitation.
- **Figure A.4: Average Steps (Last 25 Episodes) vs. Episode.** Illustrates that PPO maintains stable performance, with a lower average number of steps compared to Dueling DQN, suggesting a more consistent policy.

4 Preprocessing and Exploration-Exploitation

Both PPO and Dueling DQN used the `RGBImgPartialObsWrapper` for 56×56 RGB partial observation image. PPO uses the raw images as input, without any preprocessing. While Dueling DQN converts the last three frames to grayscale and stacks them into a $56 \times 56 \times 3$ tensor. This enhanced temporal awareness, helping the agent retain short-term memory and better understand dynamic changes in the environment, leading to improved decision-making (See Notebook).

The PPO policy naturally balances exploration and exploitation, using only the standard environmental reward. In contrast, Dueling DQN employed an ϵ-greedy exploration strategy with ϵ-decay to progressively transition from exploration to exploitation. Additionally, Dueling DQN incorporated a reward shaping mechanism to encourage meaningful interactions within the environment. The agent received positive rewards for opening doors and actively searching for them while being penalized for closing doors unnecessarily or executing irrelevant actions, such as toggling without observable changes. This structured approach helped guide the agent toward efficient exploration and reduced wasted actions, ultimately increasing reward distribution and improving performance.

5 Evaluation and Reporting

5.1 Strengths and Weaknesses

PPO provided stable learning with minimal tuning and performed well in partially observable environments, but it required more training steps. Dueling DQN converged faster initially and benefited from experience replay but was unstable in larger environments, requiring careful tuning and reward shaping to function effectively.

5.2 Challenges and Solutions

- **Sparse Rewards:** PPO's stochastic policy aided exploration, while Dueling DQN required reward shaping.
- **Partial Observability:** PPO adapted well; Dueling DQN needed stacked frames to improve decision-making.
- **Hyperparameter Tuning:** PPO's clip ratio (0.2) and batch size adjustments improved stability; Dueling DQN required fine-tuning replay buffer and epsilon decay.

5.3 Lessons Learned

- PPO excels in sparse-reward, partially observable environments, while Dueling DQN thrives in structured tasks.
- Proper tuning is crucial: PPO benefits from balanced clipping, while Dueling DQN requires fine-tuned replay buffers and weights update frequency.
- Progressive environment scaling enhances generalization.

6 Conclusions

PPO proved more robust for MiniGrid-MultiRoom-N6-v0, balancing stability and generalization. Dueling DQN learned quickly but required extensive tuning and reward shaping. In sparse-reward, partially observable environments, policy gradient methods like PPO offer a clear advantage.

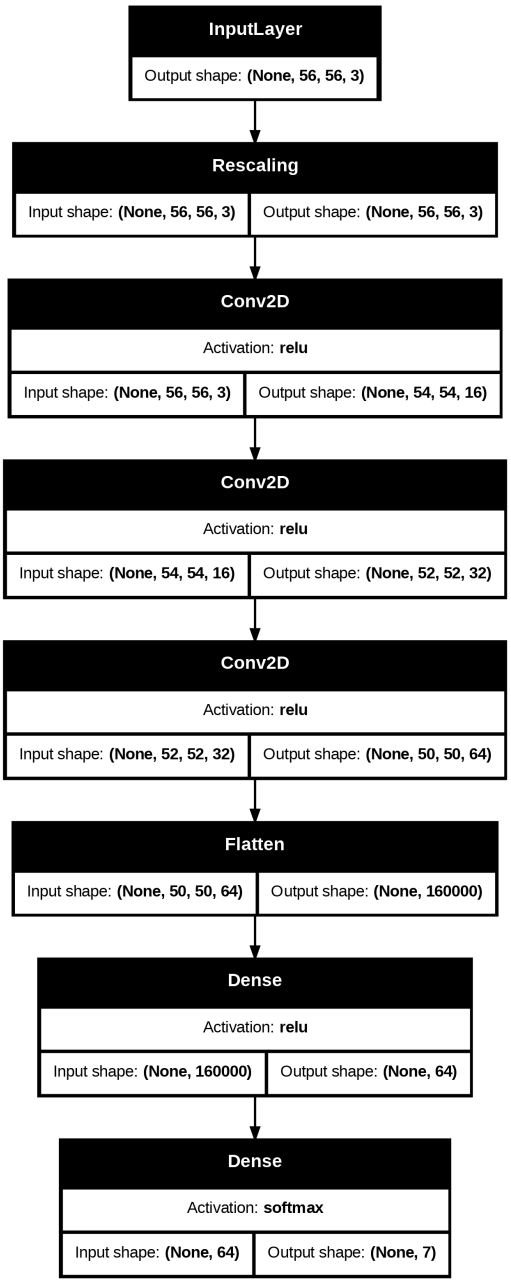
A Appendix

A.1 Notebooks

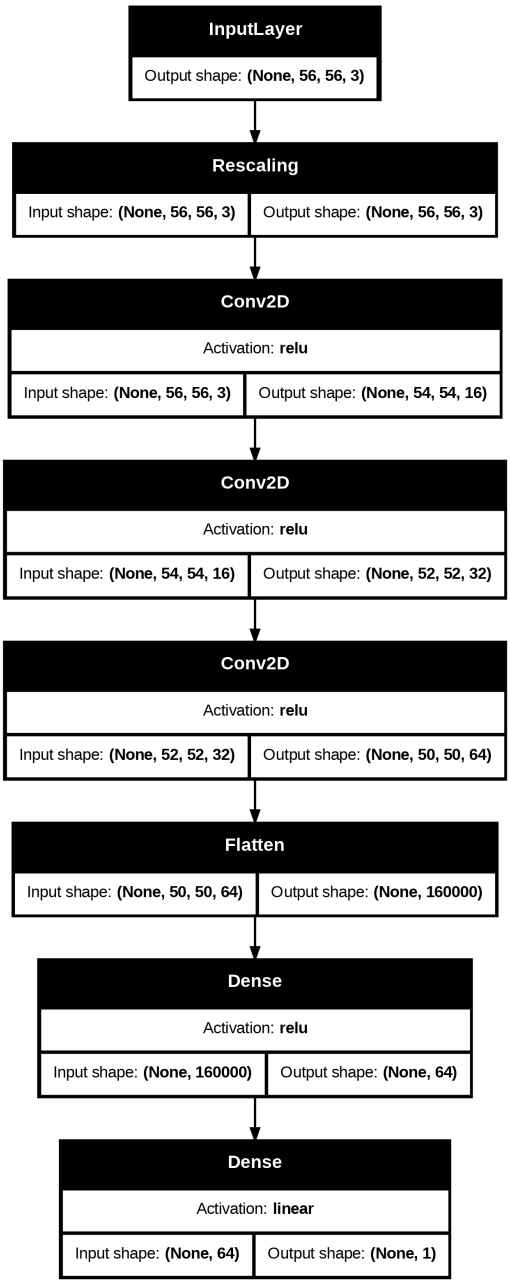
- Train Notebook
- Evaluation Notebook

A.2 Illustrations

PPO Architecture



(a) Actor Architecture. Outputs policy probabilities $\pi_\theta(a|s)$ (7 dimension vector).



(b) Critic Architecture. Outputs value for given state.

Figure A.1: PPO Architecture (See Notebook).

Dueling DQN Architecture

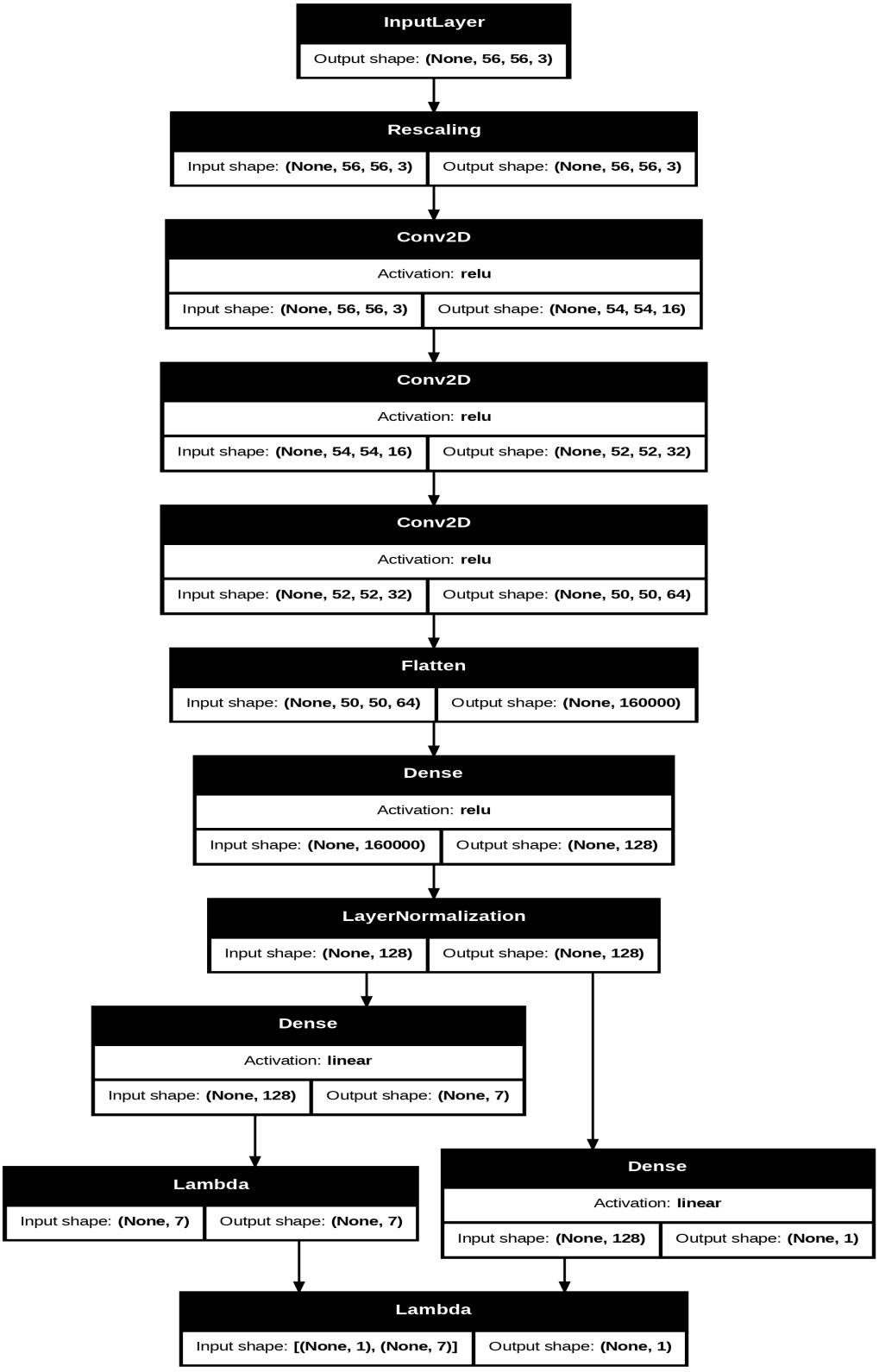


Figure A.2: Dueling DQN Architecture. Outputs two variables. First is $Q(a|s)$ which in our case is a 7 dimension vector per state, and $V(s)$ which is a scalar per state (See Notebook).

Performance Metrics

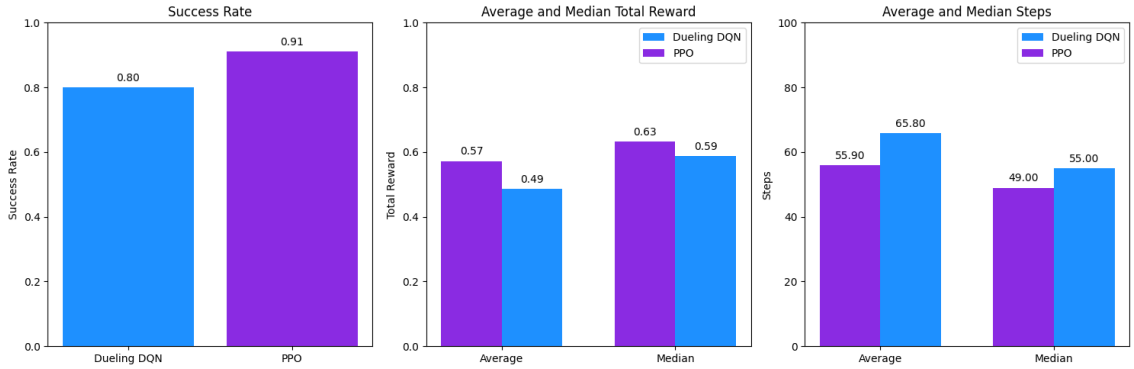


Figure A.3: Performance metrics for PPO and Dueling DQN. Left graph shows the success rate for both models given 100 episodes. An episodes succeed if it *done* and not *truncated*. Middle graph shows the average and median rewards of each model for 100 episodes. Right graph shows the average and median steps per 100 episodes.

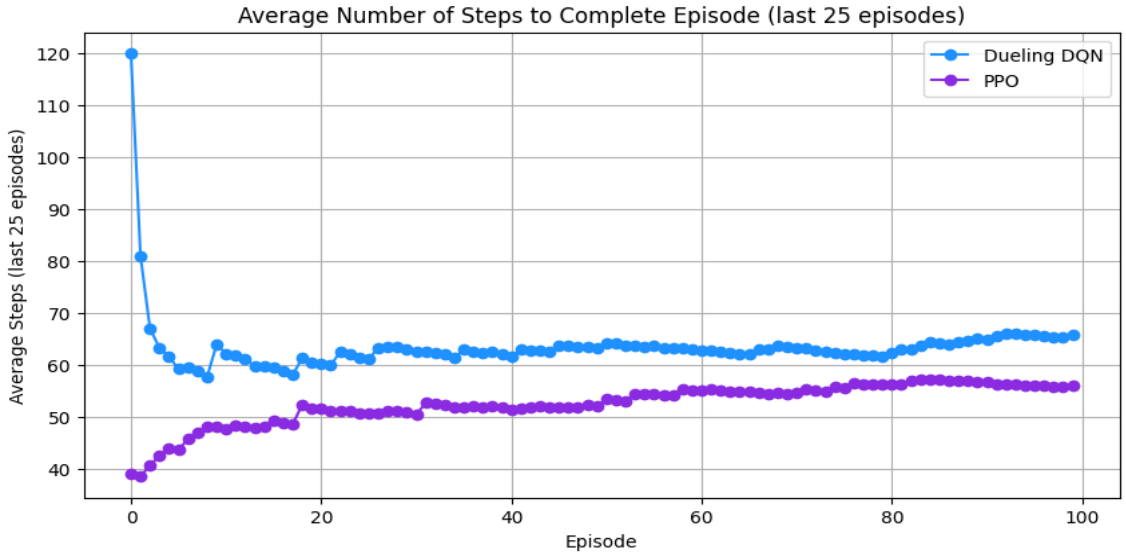


Figure A.4: Average Step of last 25 episodes vs Episode for PPO and Dueling DQN.

Agents Training Visualization Metrics

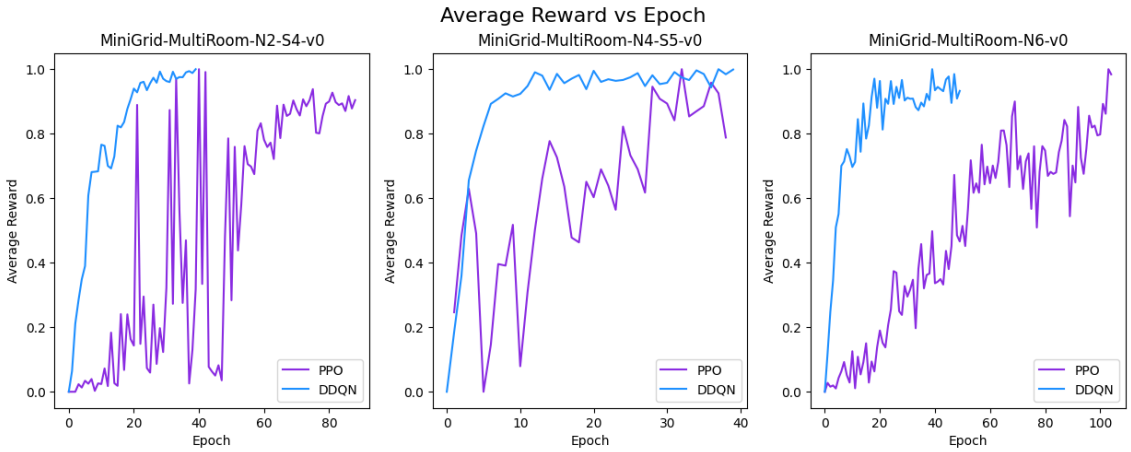


Figure A.5: PPO and Dueling DQN Average Reward vs Epoch for all environments. The reward values are normalized, as Dueling DQN applies reward shaping, leading to higher reward values compared to PPO (see Sec. 2).

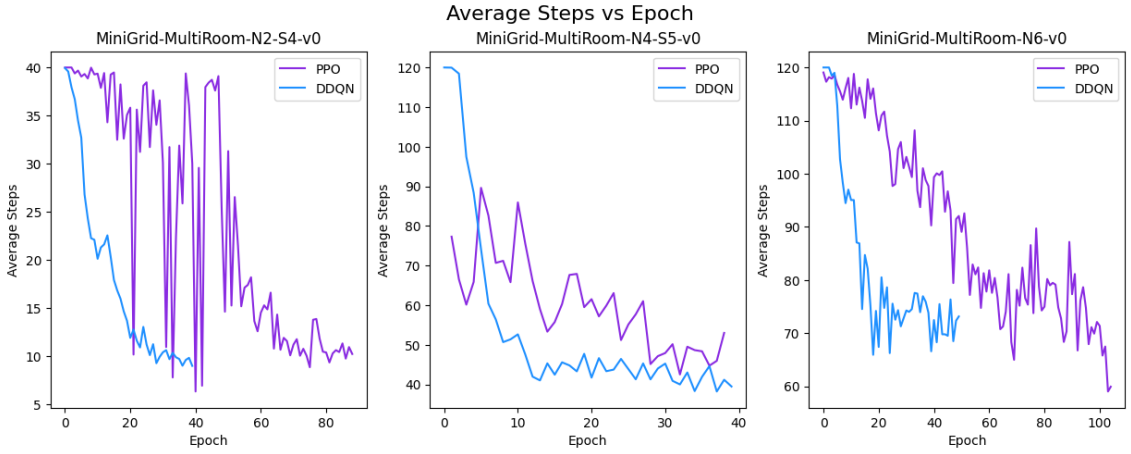


Figure A.6: PPO and Dueling DQN Average Steps vs Epoch for all environments.

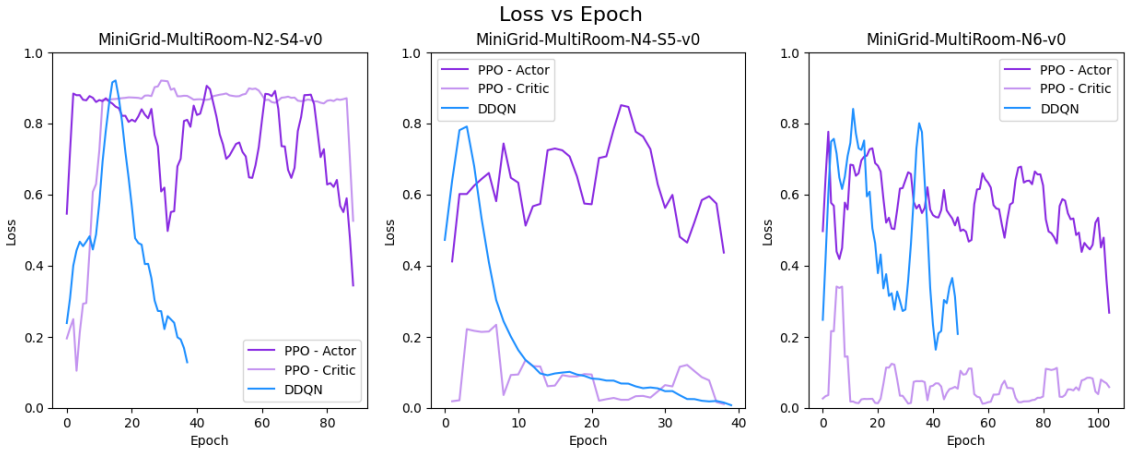


Figure A.7: Dueling DQN, PPO-Actor, and PPO-Critic Loss vs Epoch for all environments. The loss values for all models are normalized for consistency. A moving average of 5 epochs is applied to smooth out fluctuations.

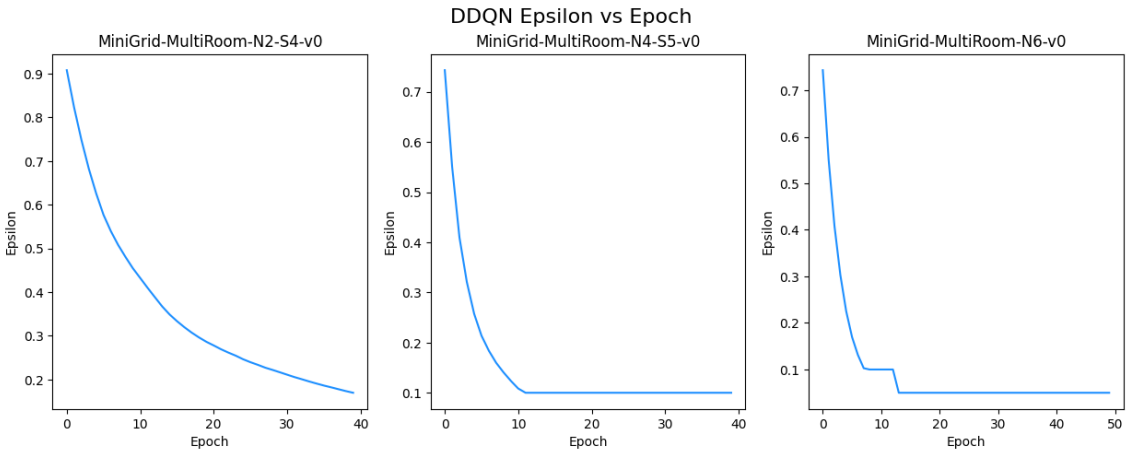


Figure A.8: Dueling DQN ϵ vs Epoch for all environments.