Comparative Analysis of Enhanced DQN and PPO Algorithms for Reinforcement Learning

Lathitha Nongauza
Student ID: 2615978

***Abstract*—This paper presents a comparative analysis of two reinforcement learning algorithms: Deep Q-Network (DQN) and Proximal Policy Optimization (PPO). While DQN serves as a foundational value-based method, PPO represents an advanced actor-critic approach with inherent stability mechanisms. We investigate the effects of strategic enhancements on both algorithms, focusing on training stability, sample efficiency, and policy optimization in complex environments. Our modifications include frame stacking for temporal awareness, biologically-inspired memory management for DQN, and residual connections for PPO, demonstrating significant improvements over vanilla implementations. The code is available at: https://github.com/Laths1/RL-project/tree/main.**

I. Introduction

Reinforcement learning (RL) has demonstrated remarkable success in solving complex sequential decision-making problems. Among the diverse RL algorithms, Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) represent two fundamentally different approaches to learning optimal policies. DQN, as a value-based method, has been extensively studied in academic settings and serves as our in-class reference algorithm. In contrast, PPO extends beyond traditional curriculum coverage as an advanced actor-critic method that directly optimizes policy parameters.

This research conducts a comprehensive comparison between enhanced versions of both algorithms, examining how strategic modifications address common challenges in training stability, sample efficiency, and policy optimization. By implementing systematic improvements to both baseline algorithms, we provide insights into the relative strengths and limitations of value-based versus policy-based approaches in complex environments with high-dimensional state spaces.

II. Background: Proximal Policy Optimization

Proximal Policy Optimization (PPO) belongs to the policy optimization family of reinforcement learning algorithms and builds upon the actor-critic architecture [1]. Unlike value-based methods like DQN that learn action-value functions, PPO directly optimizes the policy while maintaining training stability through sophisticated constraint mechanisms.

A. Clipped Surrogate Objective

The core innovation of PPO lies in its clipped surrogate objective function, which constrains policy updates to prevent destructively large changes. This is mathematically formulated as:

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

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ represents the probability ratio between new and old policies, $\hat{A}_t$ is the estimated advantage, and ϵ is a hyperparameter defining the clipping range. This mechanism ensures policy updates remain within a trusted region, preventing the rapid policy divergence that often plagues other policy gradient methods.

B. KL Divergence Regularization

Complementing the clipped objective, PPO utilizes Kullback-Leibler (KL) divergence as both a regularization term and an early stopping criterion. The KL divergence between old and new policies is defined as:

$$D_{KL}(\pi_{\theta_{old}} \| \pi_\theta) = \mathbb{E}_{s \sim \rho_{\pi_{\theta_{old}}}} \left[\sum_a \pi_{\theta_{old}}(a|s) \log \frac{\pi_{\theta_{old}}(a|s)}{\pi_\theta(a|s)} \right] \tag{2}$$

This provides a natural distance metric for policy updates, allowing PPO to maintain stable learning while efficiently utilizing collected experience through multiple optimization epochs.

III. Methodology

Our experimental methodology begins with implementing vanilla versions of both DQN and PPO as baseline comparators. Both baseline implementations utilize convolutional neural network (CNN) feature extractors for processing visual inputs, with all subsequent enhancements built upon these foundations. The baseline DQN agent was configured with a learning rate (α) of 0.05, a discount factor (γ) of 0.99, and a target network synchronization frequency of every 5 episodes. The baseline PPO agent used a learning rate of 2.5×10^{-4}, a discount factor (γ) of 0.99, a Generalized Advantage Estimation parameter (λ) of 0.95, and a clipping epsilon (ϵ) of 0.2 for its surrogate objective.

A. DQN Enhancements

1) Frame Stacking for Temporal Awareness: The vanilla DQN processes individual frames as independent states, which fails to capture the temporal dependencies crucial for dynamic environments. We implement frame stacking with $N = 4$ consecutive frames, creating composite states that provide motion context and enable the agent to perceive object dynamics and directional information. Formally, the stacked state at time t is defined as:

$$S_t = \{f_t, f_{t-1}, f_{t-2}, f_{t-3}\} \tag{3}$$

where f_t represents the frame at time t. This approach more closely resembles human perception, where actions are based on sequences of observations rather than isolated moments.

2) Biologically-Inspired Memory Management: Traditional experience replay buffers employ uniform sampling, treating all experiences equally regardless of their potential learning value. We introduce a sophisticated memory system inspired by human cognitive processes, where memories vary in significance and persistence (Algorithm 1). The enhanced DQN agent utilized a significantly larger memory buffer of $100,000$ transitions and was trained with a lower, more stable learning rate of 0.001.

Our implementation introduces two key metrics: an *uncertainty measure* based on temporal difference (TD) errors between Q-values and target Q-values, and an *age tracker* that increments each episode. The sampling probability for transition i is determined by:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{4}$$

where p_i represents the priority of transition i, and α controls the degree of prioritization. The sampling distribution balances three components, governed by the hyperparameters uncertainty_dist = 0.5 and age_dist = 0.3:

- **Uncertainty-prioritized samples**: Emphasizing transitions with high TD errors
- **Age-prioritized samples**: Preserving valuable older memories
- **Random samples**: Maintaining diversity and preventing mode collapse

When transitions are sampled, their age resets to zero and uncertainty values recompute, creating a self-organizing buffer where valuable memories persist while less useful ones naturally fade. This process mirrors human memory consolidation and forgetting mechanisms. For computational efficiency, updates occur based on an age threshold (age_update = 10) rather than every episode, and uncertainty estimates decay over time (uncertainty_decay = 0.05). The mini-batch size for learning was set to 32.

3) Training Stabilization Techniques: To address DQN's known instability issues, we implement reward clipping to constrain value ranges and gradient clipping to prevent explosive updates. The target network update frequency was also increased to 1000 episodes to enhance learning stability. Gradient clipping is applied as:

$$\text{grad} = \text{grad} \cdot \min\left(1, \frac{\theta_{\max}}{\|\text{grad}\|}\right) \tag{5}$$

ensuring more consistent and reliable training progression.

B. PPO Enhancements

1) Temporal Processing via Frame Stacking: Consistent with our DQN improvements, we integrate the same frame stacking mechanism ($N = 4$) into PPO, enabling the policy network to leverage temporal information for more informed action selection.

2) Residual Connections for Enhanced Gradient Flow: We replace standard convolutional layers with residual blocks in the feature extractor. These blocks incorporate skip connections that bypass nonlinear transformations, creating gradient highways that mitigate vanishing gradient problems in deeper networks. The residual block operation is defined as:

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{6}$$

where x and y are the input and output vectors, and $\mathcal{F}(x, \{W_i\})$ represents the residual mapping to be learned. This architectural improvement allows for more effective feature learning across multiple abstraction levels, from low-level visual patterns to high-level strategic representations.

3) Deep Residual Connections and Training Stability: We further improved upon this through a deeper CNN architecture, inspired by the ResNet architecture. The enhanced PPO agent was configured with a lower learning rate of 10^{-4} and incorporated an entropy coefficient of 0.01 to encourage exploration. The encoder is structured into four hierarchical stages that progressively abstract spatial information while expanding the feature repertoire. The process begins with an initial 7×7 convolution and max-pooling layer to aggressively reduce the input dimensionality. This is followed by the sequential stages of residual blocks: Stage 1 contains 3 blocks with 64 channels, Stage 2 contains 4 blocks with 128 channels, Stage 3 contains 6 blocks with 256 channels, and Stage 4 contains 3 blocks with 512 channels. The first block in Stages 2, 3, and 4 employs a stride of 2, systematically halving the spatial resolution while doubling the number of feature maps.

This feature extractor is integrated end-to-end with the agent's policy and value heads. Following the convolutional stages, we employ global average pooling to condense the final 3×3 feature maps into a robust 512-dimensional vector, which is then projected to a final feature representation via a fully connected layer. The resulting feature vector is fed in parallel to a linear actor network, which outputs a softmax policy over the action space, and a linear critic network, which estimates the state-value function. To ensure training stability, we employ several key techniques: the global norm of gradients is clipped to a maximum value of 0.5, the advantages are normalized before the PPO update, and the learning rate is annealed using a cosine schedule. Other parameters remained consistent with the baseline, including epochs $= 4$, minibatch_size $= 64$, batch_size $= 2048$, and a value function coefficient of 0.5.

IV. Experimental Results

The results present the performance of the models averaged over 5 runs for statistical significance. The figures have been

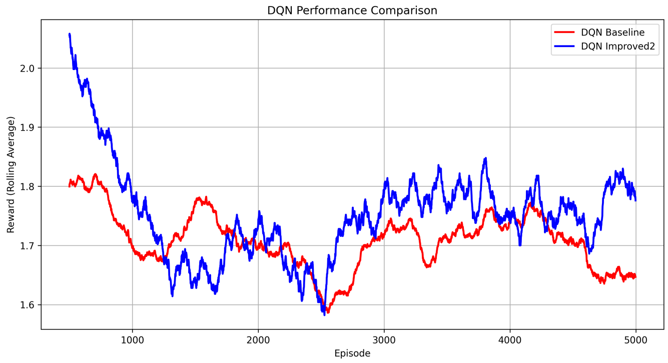

Fig. 1. DQN performance comparison over 5000 episodes, showing baseline instability and improved agent's initial performance gains.

smoothed using a rolling average. We are primarily concerned with the trend of the rewards.

A. Deep Q-Network (DQN)

1) Baseline Performance Analysis: The baseline DQN implementation exhibits significant instability in its reward returns, converging to an average episodic reward no greater than 1 (Figure 1). This indicates a fundamental failure to learn policies that effectively exploit the primary reward sources within the environment. Consequently, the agent's achievement rate remains negligible. Although the baseline agent engages with the environment for extended periods, as evidenced by its high average episode length, this interaction is inefficient. This is reflected in its performance metrics, which show a mean number of unique achievements near zero and a mean reward of approximately -1(Figure 5).

2) Enhanced DQN Agent: The improved DQN agent demonstrates a marked performance increase during initial training phases, achieving substantially higher reward returns. However, over extended training epochs, the agent's reward converges to a level comparable to the baseline, suggesting a potential limitation in long-term policy refinement (Figure 1). Despite this similarity in aggregate reward, a granular analysis of in-game achievements reveals a superior policy quality. The improved agent secures a significantly higher success rate for targeted objectives, such as *collect sapling* and *wake up*(Figure 6).

Furthermore, the enhanced agent operates with greater efficiency, completing episodes in fewer steps, as indicated by its lower average episode length(Figure 5). This improved sample efficiency is coupled with superior overall performance, as measured by a higher mean reward, a greater number of unique achievements unlocked, and a higher total achievement count. These results suggest that the modifications to the baseline DQN successfully promote the learning of more directed and effective policies.

B. Proximal Policy Optimization (PPO)

1) Baseline PPO Performance: The baseline PPO implementation demonstrates significantly stronger performance than the DQN baseline, consistently achieving rewards above a value of 3. As shown in Figure 2, the upward trend in the learning curve suggests that with additional training time, the agent's policy would likely continue to improve. However, this potential was not fully realized due to computational constraints. An analysis of the agent's behavior confirms that it successfully learns a meaningful policy, reliably unlocking a suite of achievements including *collect drink*, *collect sapling*, *collect wood*, *defeat zombie*, *make wood sword*, *place tree*, and *place table*.

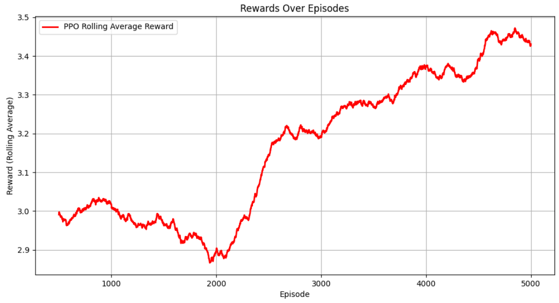

Fig. 2. Baseline PPO performance showing upward trend and potential for further improvement.

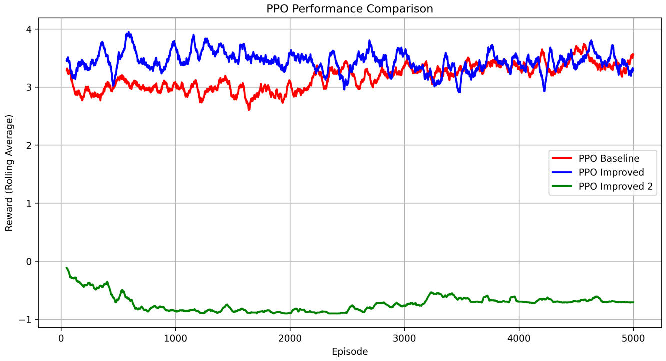

Fig. 3. PPO performance comparison over 5000 episodes, demonstrating the improved agent's consistent superiority.

2) Architectural Improvements: Our investigation began by integrating a CNN with residual connections into the PPO framework. This initial modification (Figure 3) yielded performance comparable to the baseline, indicating that this change alone was insufficient to produce a significant gain. To probe the limits of architectural complexity, we subsequently replaced the CNN with a deeper variant that also included residual connections. This deep CNN agent failed to learn an effective policy entirely, consistently receiving negative rewards and highlighting the optimization challenges of very deep networks in this environment.

The final improved model, which builds upon the initial residual CNN with further algorithmic refinements, demonstrates a clear superiority over the baseline. Crucially, it unlocks all the achievements of the baseline agent and also secures the additional achievement of *make wood pickaxe*(Figure

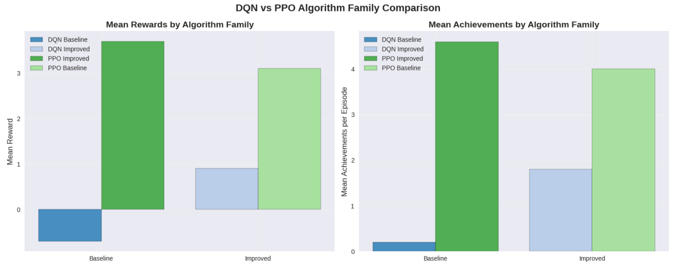

Fig. 4. Direct comparison of DQN vs PPO reward performance, establishing PPO as the more effective algorithm.

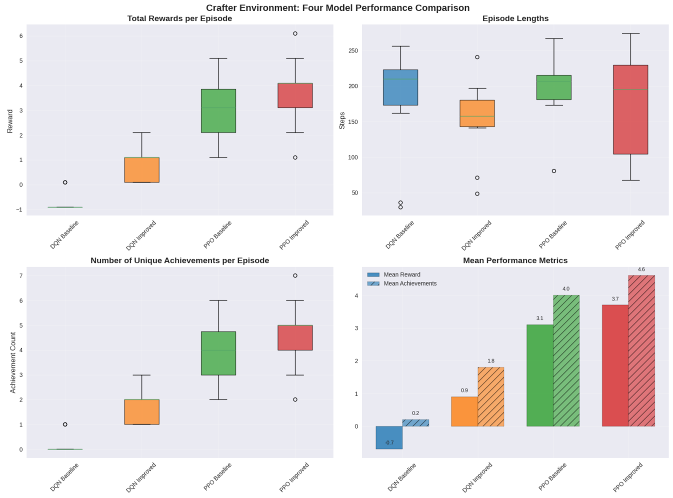

Fig. 5. Comprehensive game statistics comparison across all experimental conditions.

7). This expanded capability is reflected in its quantitative metrics, which show a higher average episode reward, a greater number of unique achievements per episode, and superior overall mean reward and achievement counts. A detailed analysis of achievement success rates reveals that the improved model outperforms the baseline on four of the nine achievements, matches its performance on three, and is outperformed by the baseline on the remaining two.

V. Discussion

The experimental results presented in Section IV offer insights into the challenges and opportunities of deep reinforcement learning in complex, sparse-reward environments like *Crafter*. Our findings highlight the distinct performance characteristics of different algorithms and the nuanced impact of architectural choices.

A. Algorithmic Considerations: DQN vs. PPO

A primary observation is the stark performance differential between the baseline DQN and PPO agents. The DQN's instability and failure to surpass an average reward of 1 underscore its limitations in this domain. As an off-policy algorithm, DQN is highly reliant on the quality of its experience replay, and its maximization bias can lead to overestimation of Q-values, resulting in the observed unstable learning and poor convergence. In contrast, PPO's on-policy, policy-gradient-based approach provides a more stable and guided learning process. Its inherent tendency to take smaller, more conservative policy updates allowed it to reliably explore and master a foundational set of tasks, achieving significantly higher rewards and a diverse set of achievements. This establishes PPO as a more robust and effective starting point for this class of environments (Figure 4).

B. The Impact of Architectural Choices

The architectural experiments within the PPO framework reveal that increased model complexity is not a panacea. The initial integration of a residual CNN yielded performance on par with the baseline, suggesting that the representational capacity of the original network was sufficient for the set of achievements the baseline had mastered. However, the complete failure of the deeper CNN to learn any positive-reward behavior is a critical finding. It indicates that simply increasing depth introduces optimization challenges, such as vanishing gradients or difficult credit assignment, that the PPO algorithm in its standard form cannot overcome. This underscores the necessity of careful architectural design and the potential need for specialized training techniques when scaling network depth.

C. Qualitative vs. Quantitative Performance

A key nuance in our results is the distinction between aggregate reward and specific achievement unlocks. The improved DQN's final reward converged to a low level similar to its baseline, yet it achieved a higher success rate in specific tasks like *collect sapling* and *wake up*. Similarly, the final improved PPO model, while superior in all quantitative metrics, did not uniformly outperform the baseline on every individual achievement. This demonstrates that the overall reward signal, while useful for tracking general progress, can mask important qualitative differences in agent behavior. An agent can learn a more efficient or diverse policy—as evidenced by the PPO agent unlocking the *make wood pickaxe* achievement—without this being fully reflected in a short-term increase in the aggregate reward metric. This argues for the use of multi-faceted evaluation criteria, including achievement-based metrics, to properly assess agent capability.

VI. Conclusion and Future Work

In conclusion, while PPO proves to be a more effective backbone than DQN for this task, our work shows that successful learning is contingent on a careful co-design of algorithms and architectures. Future work will focus on addressing the observed limitations, particularly the instability of DQN and the optimization challenges of deep CNNs. Promising directions include integrating advanced techniques from the DQN literature (e.g., Rainbow) to stabilize training, and employing network architectures specifically designed for RL, such as Impala CNNs, to enable stable training of deeper models without compromising performance.

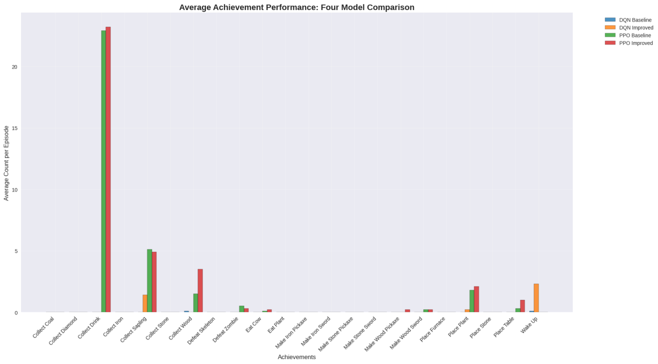

Fig. 6. Average achievement count comparison showing improved PPO agent's superior performance.

References

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

[2] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

Appendix: Sampling Algorithm

Algorithm 1 Biologically-Inspired Experience Sampling

```
procedure Sample($batch\_size$)
    $n \leftarrow |buffer|$
    if $n = 0$ then return $[\,], [\,], [\,]$
    end if
    decay_uncertainties()
    Compute $n_p, n_a, n_r$ from sampling distribution
    $indices \leftarrow \emptyset$
    if $n_p > 0$ then {High TD-error samples}
        Sample $n_p$ indices proportionally to priorities
        $indices \leftarrow indices \cup$ sampled
    end if
    if $n_a > 0$ then {Oldest memories}
        Select $n_a$ indices with maximum age
        $indices \leftarrow indices \cup$ selected
    end if
    if $n_r > 0$ then {Random diversity}
        Sample $n_r$ random indices from remaining
        $indices \leftarrow indices \cup$ random
    end if
    if $|indices| < batch\_size$ then {Fill batch}
        Add random samples from remaining indices
    end if
    Truncate $indices$ to $batch\_size$
    return $batch, indices, ages$
end procedure=0
```

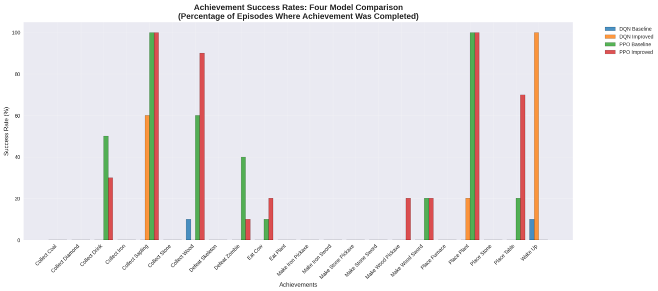

Fig. 7. Average achievement rate comparison highlighting nuanced performance differences.