Enhancing Proximal Policy Optimization with Code-Level and OPPO+ Optimizations for Pathfinding in the Minigrid Environment.

Jay Gandhi

Abstract

This paper is an exploration of enhancements to the Proximal Policy Optimization (PPO) reinforcement learning algorithm for improved pathfinding performance in the 2D MiniGrid environment. While PPO has been proven to be extremely effective for general reinforcement learning tasks, its application in specific pathfinding situations can be further optimized to achieve more efficient and accurate results. This study presents a dual enhancement strategy that integrates code-level improvements with the OPPO+ algorithm to refine PPO's performance. The proposed approach is a modified PPO algorithm designed to improve exploration and reward efficiency, in the context of the MiniGrid 2D environment where an agent is required to navigate varying maps to reach the exit. The code-level optimizations involve modifications to network architecture, parameter tuning, and policy adjustments, which work to stabilize learning and improve convergence. Meanwhile, the OPPO+ concept is applied to encourage exploration techniques, such as multi-batch training and bonus reward scaling. These enhancements, in turn, assist the agent in gaining a comprehensive understanding of the environment and ability to navigate complex grid layouts. Through this dual enhancement approach, the PPO model demonstrates gains in efficiency, as shown through the benchmarks and performance metrics. Overall, the experiments conducted reveal improved PPO capabilities in complex environments, which provides valuable insights for the field of reinforcement learning-based pathfinding.

Code — https://github.com/JayG17/Optimized-PPO-Research

1. Introduction

Reward-based reinforcement learning is a core concept that is continuously being researched and improved in the field of machine learning. The Proximal Policy Optimization (PPO) algorithm is a key element in this field and is popular for its balanced approach to exploration and exploitation, allowing agents to adapt in complex scenarios. However, when applied to specific tasks like pathfinding in 2D grid environments, where adaptability and long-term decision-making are essential, the standard PPO implementation often falls short, facing limitations. This underperformance is caused by a need for more efficient exploration and tailored reward optimization for the specific setting, in this case the Minigrid environment robot-to-exit task.

The Minigrid environment is a modular 2D platform designed to help researchers implement, develop, and visualize machine learning algorithms in the context of reinforcement learning. It provides a structured setting where researchers can develop goal-oriented tasks and measure an agent's performance. The environment tests and evaluates an agent's ability to explore effectively, recognize useful patterns, and make optimal decisions that maximize rewards. Through this, Minigrid serves as an important tool for advancing research in reinforcement learning, particularly in navigation, exploration, and decision-making of dynamic environments.

This project addresses these performance gaps by enhancing PPO through a dual approach, combining code-level optimizations with the OPPO+ algorithm. The code-level improvements focus on refining the key parameters and structure of the algorithm to make learning faster and converge better. This fine-tuning helps the agent adapt efficiently during and throughout episodes. Furthermore, the OPPO+ concept provides new techniques for promoting exploration, like multi-batch training and a bonus reward scaling feature. Through these additions, the agent can develop a heightened understanding of the environment, obstacles, and pathing.

The following sections will explore the specific enhancements made to PPO, focusing on how the code-level optimizations and OPPO+ modifications work. This will include a detailed analysis of how the adjustments contribute to the agent efficiency and overall performance in the Minigrid setup. The basic PPO model will serve as a baseline, comparing it across four total phases. Phase 1 represents the unmodified and basic PPO, phase 2 incorporates code-level optimizations, phase 3 introduces OPPO+, and phase 4 combines both code-level and OPPO+ enhancements. Key performance metrics like reward collection and path efficiency will highlight progress within each phase, showing impact of the additions. Ultimately, this structured approach will show how combining these optimizations can advance reinforcement learning ini pathfinding and decision-making. This will, in turn, hopefully provide extremely useful insights for the RL field and PPO research.

2. Related Work

This project expands on various concepts in reinforcement learning, particularly in the field of Proximal Policy Optimization (PPO) and its application to navigation tasks. This

section examines relevant research on the Minigrid environment, traditional PPO setups, code-level optimizations for PPO, and the OPPO+ concept, which all serve as the foundation for the dual enhancement strategy implemented for this project and research.

2.1 Minigrid Environment Documentation

The paper, *Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks* [1], introduces the Minigrid testing environment. This is a toolkit designed for researchers to create and customize 2D grid-based settings for testing reinforcement learning algorithms. MiniGrid provides a streamlined way to design environments with agents, obstacles, and target goals, making it perfect for studying navigation and exploration tasks. With an emphasis on "modularity" as a key advantage, this suit is ideal for benchmarking the needs of this project across different experimental phases. For this project, MiniGrid is the primary testing environment for the enhanced PPO algorithm. Its comprehensive documentation [2] allowed for straightforward customization to meet this study's overall goals. Using the built-in methods from the documentation, I was able to develop the environment for the robot-to-exit scenario, while easily capturing performance gains and key benchmarks. This project builds on Minigrid's capabilities and available documentation to generate a structured navigation task. In all, Minigrid's modular structure and open documentation was essential for the benchmarking required of this study.

2.2 Existing Proximal Policy Optimization Library

Proximal Policy Optimization is one of the most popular RL algorithms for its ability to learn while maintaining a balance between stability and performance. PPO achieves this stability by limiting policy updates through a clipping technique. The Stable Baselines3 reinforcement learning library [3] provides a basic, yet powerful implementation of PPO for simple experimentation. This library allows for phase 1 in this study, which is the PPO baseline benchmark to be used for comparison with later phases. By clearly establishing the standard PPO performance in the Minigrid environment setup, we can clearly identify the effect of the code-level optimizations and OPPO+ modifications.

2.3 Research on Code-Level Optimizations

The paper, *Implementation Matters in Deep Policy Gradients: A Case Study on PPO and TRPO* [4], discusses code-level optimizations. The article underlines how these code-level optimizations are seemingly small changes to the code structure, however can have large positive impacts on agent performance. "These optimizations may appear as merely surface-level or insignificant algorithmic changes to the core policy ... [but] we find that they dramatically affect the performance of PPO." The following are important concepts that helped to guide my phase 2 (code-level optimizations) implementation:

- Orthogonal Initialization: Orthogonal initialization sets weights to be orthogonal/perpendicular to each other in the network, which in turn helps to stabilize weight scaling when training. This is used instead of the standard weight initialization methods, like Gaussian distribution, which can possibly make weights in random or arbitrary directions.
- Hyperbolic Tan Activations: This concept sets the activation function as the hyperbolic tan (Tanh) function. Essentially, this scales the output between -1 and 1. This bound helps maintain stability for smoother model learning and training. It promotes nice and stable gradients in the network.
- Adam Learning Rate Annealing: Basically, Adam (Adaptive Moment Estimation) Learning Rate Annealing is a concept where the learning rate is decreased as training continues, so that the model converges to the already learned knowledge. This is essentially a concept to promote some exploitation for fine-tuning the policy. With PPO and navigation, this is critical to allow precise decision-making even when heightened exploration is occurring.
- Reward Clipping with Observation Normalization: This concept involves clipping rewards and observations to prevent outliers and other extreme values from influencing the learning process. This is an additional measure to help PPO with balancing for stability and performance, ensuring that learning occurs with actually useful information. Furthermore, observation normalization is simply normalizing the observable data, or making the known results of actions more consistent. Together, these enhancements make the agent have smooth and reliable learning progress.
- Global Gradient Clipping: This technique is to set a 0.5 global norm or "norm of the concatenated gradients of all parameters" [4]. This is a reduction of abrupt shifts in any specific policy parameter, forcing more steady and stable progress.
- Policy and Value Network Architecture: A two layer (64 unit by 64 unit) structure is implemented for the pi (policy network) and the vf (value function) which helps the model to recognize patterns. Each network has two layers of 64 units, where the policy network decides actions, and the value function evaluation actions. This yields a deeper comprehension of grid states for the agent, helping with navigation.

2.4 Research on OPPO+ Algorithm

As outlined in the article, *A Theoretical Analysis of Optimistic Proximal Policy Optimization in Linear Markov Decision Processes* [5], OPPO+ is an advanced extension of the basic PPO approach for reinforcement learning. This enhancement introduces more balanced exploration and exploitation so that the agent can learn more in potentially complex environments. The paper highlights core OPPO+ concept modifications that assist in developing the improved structure. The following are important concepts that helped to guide my phase 3 (OPPO+ Algorithm) implementation:

- Multi-batched Updating Mechanism: A main feature of OPPO+ is a multi-batch approach. Essentially, this is where the agent updates the policy after analyzing performance of multiple data batches. For example, if the agent takes 1,000 steps and gathers observations, then the steps are grouped together into a batch for processing.
- Reward Scaling/Clipping and Bonus Functions: OPPO+ involves scaling rewards, with clipping, to follow a more consistent range for steadier policy updates. Furthermore, the concept of bonus functions is developing a relevant method to apply a bonus. My interpretation of this was a bonus method for exploration, which involves scaling earlier reward values for early exploration, with less scaling as progression occurs.

3. Problem Definition

The main challenge of this project is to improve the overall performance of the Proximal Policy Optimization algorithm for complex pathfinding tasks in 2D environments with limited observation range. These tasks require adaptability and real-time decision making, since agents must navigate obstacles, maximize rewards, and minimize steps. PPO offers a balance between exploration and exploitation, but underperforms with such demanding navigation tasks. The standard PPO method is a good baseline, however additional enhancements can be applied to maximize its effectiveness in this context.

In the specific Minigrid environment setup, agents operate with limited perception. With this, agents require much adaptability and flexibility. The standard PPO algorithm isn't able to meet these needs for a few reasons. Firstly, premature convergence occurs by stabilizing policies too early. While this helps with overall stability, the agent isn't able to get a comprehensive understanding and robust policy. Secondly, the exploration incentives are not optimized in the basic PPO setup. As a result, the agent can place more emphasis on exploitation, even when exploration is needed for navigating and learning complex environments. Thirdly, the basic reward structure lacks adaptability, meaning that the agent will struggle to maximize rewards in dynamic settings. Ultimately, these listed limitations and more prevent the agent from reaching peak performance with learning and adapting in the Minigrid environment.

These points yield the research question of this project: How can the basic PPO algorithm be enhanced to achieve improved exploration effectiveness and reward consistency in grid-based navigation tasks in Minigrid? This project contributes a dual enhancement approach of innovative implementations:

1. Code-Level Optimizations: This is the first approach at enhancing the PPO algorithm with improvements through integration of orthogonal initialization, hyperbolic tan activations, adam learning rate anneals, reward clipping, and more. Such strategies are designed for stabilized training and smoother learning. These optimizations should ideally enhance the overall architecture for the agent learning process.
2. OPPO+ Algorithm Integration: The second approach involves advancing controlled exploration, with the OPPO+ concepts. This includes a multi-batched updating system, tailored reward scaling/clipping, and a custom bonus function. These additions aim to encourage exploration, mitigate the PPO premature convergence issue, and overall yield more efficient and flexible policies.

These concepts will be implemented across different phases, each involving thorough benchmarking, testing, and evaluation. Ultimately, a final combined enhancement algorithm will be developed, aiming to deliver optimal results and maximized performance.

4. Methodology

This project uses a multi-phase approach to incrementally test and enhance the Proximal Policy Optimization algorithm. This methodology includes four main stages, to focus on developing improvements for the pathfinding in the 2D Minigrid environment. Each phase involves a core concept implementation and executing tests. In short, phase 0 is the Minigrid environment setup, phase 1 is the basic PPO implementation, phase 2 is the code-level optimizations, phase 3 is the OPPO+ algorithm, and phase 4 is the enhanced and combined implementation of code-level optimizations and OPPO+.

4.1 Phase 0 - Environment Setup

The Minigrid testing environment is the foundation for this project. This is the 2D grid-based space that simulates the robot-to-exit scenario where the agent is required to reach an exit while navigating around obstacles (walls). This grid is a 10x10 layout, with the agent starting in the top left corner and reaching the goal/exit in the bottom right corner. I apply a surrounding wall to restrict the agent movement, which makes the inner movement area 8x8. Following this, I set different wall obstacles to 2.5% of grid cells for each episode for an added layer of complexity.

The agent's view is restricted to a 7x7 grid range for the live position. This introduces another layer of complexity and realism, as the agent has to adapt and make decisions on partial information like in real-world situations. To prevent aimless movement, it's given the maximum steps to take. This setup is used in each phase, to provide a consistent, realistic, and fair testing ground for each improvement. However, phase 0 code was implemented specifically to execute and visually render the environment to confirm that the setup was accurate and functioning correctly, with correct agent movement, viewing range, goal/exit objectives, and obstacle placements (Figure 1). This was a preliminary step to ensure the environment works as intended before application of enhancements.

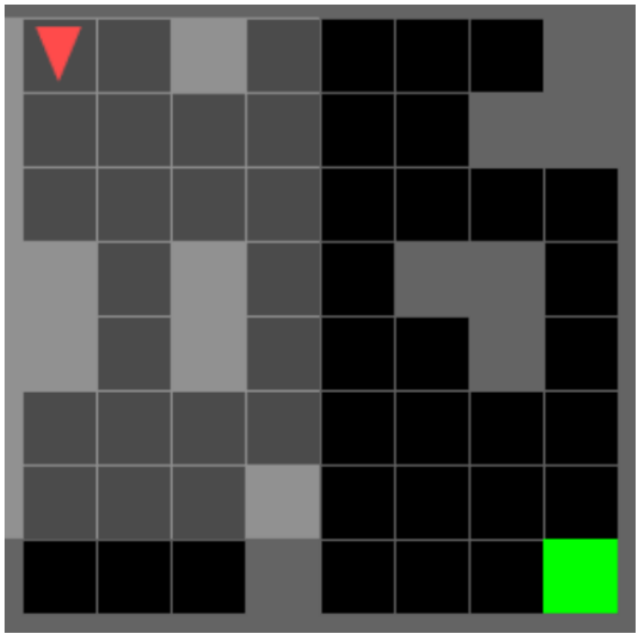

Figure 1: Visualization of the Minigrid environment, showing the 10x10 grid (with an 8x8 inner movement area) where the agent (red triangle) starts in the top-left corner, aiming to reach the goal (green square) in the bottom-right corner. The agent's "Vision" is highlighted by the light grey 7x7 grid in its forward-facing direction. Obstacles/walls (dark grey squares) and open paths form a maze-like layout, providing the agent with a navigation task.

4.2 Phase 1 - Basic PPO Implementation

In phase 1, a basic PPO model from the Stable Baselines3 library is used as a starting point. This sets the first baseline and benchmark for performance metrics to be used in comparison with later additions. As a baseline, this model relies on PPO default settings, including the following...

- Policy: MlpPolicy (multi-layer perceptron policy) is used for decision-making in the environment.
- Learning Rate: 1e-4 value for the update rate in which the model learns. This is a smaller rate, which is more stable and basic for PPO.
- n_steps: 2048 steps to collect at each batch, before performing an update. This is a common PPO amount which allows for sufficient training data.
- Clip Range: 0.2 value for clipping and stabilizing policy updates, without drastic changes.
- Gamma: 0.99 value for discount factor, which emphasizes long-term future rewards over immediate ones, a typical PPO setting.
- Entropy Coefficient: 0.01 value, which introduces small penalties for certainty. Essentially, this assists in prioritizing stability and exploitation of knowledge.

The training runs for 100,000 steps, with the model performance being assessed over 5,000 episodes. The average rewards and steps to reach that goal are recorded and act as this basic PPO baseline.

4.3 Phase 2 - Code-Level Optimizations

In phase 2, multiple code-level adjustments are applied to enhance the PPO model learning process. These changes and additions are designed to help the model with stability and adaptability, in this complex environment setup.

- Orthogonal Initialization: Through the policy configurations in the PPO setup, I initialized the network weights orthogonally. This weight setup provides smoother learning and allows the agent to exploit the known strategies in this phase.
- Hyperbolic Tan Activations: Using the Torch library, I specified the policy activation function to be Tanh. In the network layers, there is this new bound of -1 and 1 for preventing extreme values and less certain exploration.
- Adam Learning Rate Annealing: I implemented a custom learning rate annealing function to gradually decrease the learning rate over time, with a minimum threshold set at 1e-5. As training progresses, the learning rate reduces to force the model to shift from exploration to exploitation, helping it converge better.
- Reward Clipping with Observation Normalization: Using the VecNormalize wrapper and implementing clipping for rewards and observations, I aimed to enforce un-skewed training. This means that the agent identifies realistic and consistent patterns, rather than potentially misleading or skewing patterns.
- Global Gradient Clipping: I use a 0.5 gradient clip to limit the maximum gradient size in the network for the PPO model. This works by capping the gradient value at each update step, enforcing smoother and controlled policy updates.
- Policy and Value Network Architecture: The policy network (π) is used for deciding actions and the value function (vf) is used for evaluating actions. I set a two-layer architecture (64 unit by 64 unit) for the policy and value networks. This results in a deeper comprehension of the varying grid states for the agent, helping with navigation through obstacles.

Together, the code-level optimizations create an exploitation-centered PPO model that can better handle the dynamic Minigrid setup. These adjustments in phase 2 aim to achieve a baseline of increased stability and adaptability.

4.4 Phase 3 - OPPO+ Algorithm Integration

In phase 3, OPPO+ features are explored and added to the standard PPO model to encourage exploration and flexibility. Essentially, OPPO+ extends the PPO in a way that prevents premature convergence. This, in turn, promotes exploration for longer, such that it can gain a deeper understanding for complex and obstacle-filled environments, like the Minigrid setup I developed.

- Multi-batched Updating Mechanism: Instead of updating the policy after individual actions and gained observations, I structured the code to collect data in batches and analyze it before making the policy updates. This approach involves using two batches of actions in each evaluation, with totalcalculated as total_timesteps divided by the product of batch_size and multi_batch_amnt. Collecting the rewards and observations over this dual-batch structure, in contrast to individual updates per action, aims to assist the agent in developing a comprehensive understanding of the environment. Furthermore, it helps the model to avoid convergence on policies that may not be optimal.
- Reward Scaling/Clipping and Bonus Functions: To create a more stable reward process, the reward values are scaled and clipped. One formula, current_reward = np.clip(current_reward, -5, 5) / (np.std([current_reward]) + 1e-8), ensures a consistent and smooth range for the reward values. Basically, this works to prevent large skews in agent learning and, in turn, make it easier to recognize patterns that lead to the most optimal policies. Additionally, the exploration bonus function structure is applied using a bonus beta value, 0.05 * (1-(update/total_updates)). This is a progression-based bonus method that encourages early exploration. Over time, the exploration bonus is decreased to shift focus into exploitation and necessary convergence, when the proper information is gained.
  The q_rewards approach further enhances long-term objectives by factoring the future expectations. Each new_q_reward is calculated as current_reward + (model.gamma*q_rewards[-1]). This essentially takes the current (immediate) reward and adds the discounted portion of the previous step. By incorporating this progressive reward mechanism, agents aim for long-term positive outcomes, aligning with the heightened OPPO+ balance for exploration and exploitation.

The above OPPO+ strategies aim to promote an exploration process for the PPO model. Such implementations should help the agent to avoid premature convergence, and focus on continuous learning in the early stages of policy updates. Then, as training progresses the adjustments make a gradual shift back towards exploitation for finding optimal strategies after sufficient exploration occurs.

4.5 Phase 4 - Combining/Enhancing Code-Level Optimizations and OPPO+ Algorithm

Phase 4 is the final phase in this process. This phase involves applying and enhancing both the code-level optimizations (from phase 2) and the OPPO+ concepts (from phase 3). The aim is to merge the phase 2 stability-focused adjustments with the phase 3 exploration-focused implementations. Together, the goal is to develop an advanced model that is able to adapt, navigate, and perform better in the Minigrid environment, as compared to the standard PPO implementation of phase 1.

This model incorporates the orthogonal initialization, hyperbolic tan activations, learning rate annealing, reward clipping, observation normalization, policy/value network architecture, and other code-level optimizations for the stable foundation. Furthermore, the multi-batched mechanism with the reward scaling and exploration bonus function strategies are implemented for the exploration-focused OPPO+ approach. Ultimately, this phase tests whether the combined optimizations yield an effective model for the specific robot-to-exit task I developed in the Minigrid environment. Key performance metrics like total rewards, steps-to-goal, and training time all serve to help the benchmarking process.

5. Experimental Results

Phase Number and Implementation Type	Benchmark Results Averages from 3 Tests		
	Training Completion Time	Avg. Reward Value	Avg. Steps-To-Completion
Phase 1: Basic PPO	168.982 sec	8.112	69.879
Phase 2: Code-level Optimizations	453.678 sec	5.373	19.825
Phase 3: OPPO+ Implementation	307.794 sec	5.771	24.421
Phase 4: Enhanced Combination of Code-level Optimizations and OPPO+ Implementation	319.393 sec	4.949	21.606

Figure 2: Experimental benchmark results for each phase of the PPO enhancement project, showing average training completion times, reward values, and steps-to-completion, based on three tests, each consisting of 5000 episodes. Phase 1 serves as the baseline PPO, while Phases 2, 3, and 4 incorporate code-level optimizations, OPPO+ implementation, and a combined approach of both enhancements, respectively. These metrics demonstrate the impact of each enhancement on training efficiency and agent performance in the 2D Minigrid environment.

This section presents and evaluates the outcomes of the enhancements applied to the PPO algorithm across the four core phases of the project. These experiments evaluate the training completion time, average reward value, and steps-to-completion, as summarized in the above table (Figure 2). Note that each phase was tested three times over 5,000 episodes, with the results averaged to yield the metrics shown.

In phase 1, the standard PPO algorithm serves as the baseline model. It completed training in 168.982 seconds, with an average reward value of 8.112, and took an average of 69.879 steps to reach the goal. Although each navigation could ideally be completed in under 20 steps, this baseline shows a relatively high average step count to reach the goal. The training time is relatively low, suggesting that the policy stabilizes quickly but still leaves room for much improvement in terms of efficiency and goal-reaching performance.

In phase 2, code-level optimizations were introduced to enhance stability and learning efficiency. With adjustments like orthogonal initialization, reward clipping, and other refinements, notable improvements were exhibited in the steps-to-completion. The average steps dropped drastically to 19.825, highlighting significant progress from phase 1. However, this improvement came with a large trade-off that I didn't expect: training time increased to 453.678 seconds (over 2.5 times longer than in phase 1). This spike in processing time likely stems from the added complexity of the policy adjustments, which require more processing to stabilize. Furthermore, the average reward decreased to 5.373,

suggesting that while the model became more efficient in reaching the goal, the increased desire for stability limited its exploration and reward outcomes.

Phase 3 aimed to boost the model's exploration through OPPO+ concepts and incentives. By implementing features like the multi-batched updating mechanism, reward scaling, and an exploration bonus, the agent successfully avoided premature convergence (which the standard PPO models face). The average reward was similar to that of phase 2, however the point of interest is the training completion time and average steps-to-completion. This was a smaller jump in time than that of phase 1 to phase 2. Reaching completion in 307.794 seconds (less than a 2x increase from phase 1), the steps-to-completion still managed to cut down by around 65% to 24.421 average steps. This highlights the largely positive impact of the OPPO+ implementations. This is most likely due in part to the progression-based bonus method, which starts high to encourages early exploration and gradually decreases to shift focus toward exploitation as the agent learns more

In phase 4, I combined the code-level optimizations of phase 2 and the OPPO+ concepts from phase 3. Following this, I ran more tests and further refined the implementations, such that this dual-enhancement approach was optimized for the specific robot-to-exit task. The idea was to merge the benefits of stability (phase 2) and early exploration (phase 3) into an innovative and balanced solution. For the results, the average steps-to-completion was 21.606, which is a solid middle range from the previous phases. Although this wasn't the lowest step count achieved, the training time was relatively moderate at 319.393 seconds, showing a very reasonable trade-off between efficiency and effectiveness. However, it's important to note that the reward average settled at 4.949, slightly lower than in previous phases. This visible drop might indicate that the combined approach, while efficient in terms of navigation and stability, shifted focus away from the reward maximization. Ultimately, phase 4 demonstrates that by balancing stability with early exploration, the model achieves advanced performance in the complex Minigrid environment, even if it involves a slight trade-off in reward value.

Overall, the data collected in Figure 2 highlights the noticeable impacts of each phase's enhancements. Phase 1 established the baseline with a standard PPO model, yielding a solid foundation for comparison with other phases. Phase 2 focused on enhanced stability, although it required a substantial trade-off in training time. Phase 3 emphasized early exploration, achieving mid-level results for step-to-completion and rewards. And finally, phase 4 focused on finding a balance between both stability and exploration. These results reveal how specific improvements to PPO can make the model more effective at navigating a challenging 2D grid-based environment.

6. Conclusion and Future Directions

This project followed an incremental, phase-based approach to improve the Proximal Policy Optimization algorithm, specifically for enhancing model performance in a 2D Miniworld navigation task. Each phase represented a core part of this research. Phase 0 focused on the environment setup for the Minigrid environment, ensuring that all components—agent, obstacles/walls, 7x7 perception area, 10x10 grid with 8x8 agent area, objective/exit—were all correctly implemented and functioning. Phase 1 served as the baseline, demonstrating fast training but high steps-to-completion, indicating much room for improvement. Phase 2 aimed at stability and knowledge exploitation, with adjustments like orthogonal initialization and reward clipping. Such optimizations drastically reduced the steps-to-completion, at the cost of a large increase in training completion time. Phase 3 emphasized an early exploration focus with OPPO+ features, like the multi-batching mechanism and exploration bonuses. This phase saw a balance in training time and steps-to-competition, effectively preventing premature convergence. Phase 4 combined and balanced the stability-focused and exploration-focused features, for a more refined model. Ultimately, this approach yielded an innovative model that developed a comprehensive and adaptable understanding of the complex Minigrid environment.

Moving forward, it is clear that there are a variety of applications for these enhanced PPO models. Algorithms like this can be applied in reward-based navigation fields such as autonomous driving, video game AI, robots, and other machine learning areas where optimized performance and decision-making in dynamic environments are essential. Future directions include applying the dual-enhancement approach in other 2D navigation tasks available within Minigrid, or expanding to 3D navigation tasks through Miniworld (Minigrid's 3D environment extension). It's important to explore these algorithms in varied settings to continue refining for stability-focused and exploration-focused enhancements. With these next steps, future work can expand the effectiveness and applications of PPO, making it very suitable for a range of demanding RL challenges.

References

[1] M. Chevalier-Boisvert, B. Dai, M. Towers, R. Perez-Vicente, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry, *Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks*, in Advances in Neural Information Processing Systems, 2023. Available at: https://proceedings.neurips.cc/paper_files/paper/2023/file/e8916198466e8ef218a2185a491b49fa-Paper-Datasets_and_Benchmarks.pdf

[2] Farama Foundation, *Minigrid Documentation*. Available at: https://minigrid.farama.org/

[3] *Stable-Baselines3: Proximal Policy Optimization (PPO) Documentation*. Available at: https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html

[4] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, *Implementation Matters in Deep RL: A Case Study on PPO and TRPO*, in International Conference on Learning Representations, 2020. Available at: https://openreview.net/forum?id=r1etN1rtPB

[5] H. Zhong and T. Zhang, *A Theoretical Analysis of Optimistic Proximal Policy Optimization in Linear Markov Decision Processes*, in Advances in Neural Information Processing Systems, 2023. Available at: https://proceedings.neurips.cc/paper_files/paper/2023/file/e9721921b799b6ea98d37f9e77f1a7fe-Paper-Conference.pdf