VFAST Transactions on Software Engineering
https://vfast.org/journals/index.php/VTSE@ 2026, ISSN(e): 2309-3978, ISSN(p): 2411-6246
Volume 14, Number 1, January-March 2026 pp:153-174

Exploring Proximal Policy Optimization in ViZDoom: Training Agents for Complex Tasks with Hyperparameter Optimization

Areej Fatemah Meghji [1*], **Rashid Hussain** [2], **Mirza Muhammad Abbas** [1], **Abdul Lahad** [1]

[1]Department of Software Engineering, Mehran University of Engineering and Technology Jamshoro, Pakistan;
[2]Department of Cyber Security, FAST - National University of Computer & Emerging Sciences, Karachi, Pakistan

Keywords:
Reinforcement Learning, Proximal Policy Optimization, ViZDoom, Hyperparameter Optimization.

Journal Info:
Submitted:
February 10, 2026
Accepted:
March 10, 2026
Published:
March 15, 2026

Abstract ViZDoom, a Reinforcement Learning (RL) research platform based on the classic first-person shooter game Doom, allows training and evaluating RL agents across a wide range of scenarios that vary in complexity. This research trains Deep Reinforcement Learning (DRL) agents using the modern Proximal Policy Optimization (PPO) algorithm, implemented through Stable-Baselines3 (SB3), across four different ViZDoom scenarios with increasing complexity: Basic, Defend the Center, Health Gathering, and Deadly Corridor. We analyze the impact of hyperparameter tuning on the PPO algorithm in these ViZDoom scenarios using the Optuna framework. While keeping in mind the complexity of the Deadly Corridor scenario, the advanced RL techniques like reward shaping and curriculum learning are employed to achieve the objective of this scenario. The agents are evaluated using the mean episodic reward and the mean episode length as performance metrics in each scenario. The results compare the PPO agents trained on default hyperparameters with the agents trained on optimized hyperparameters in each scenario. The findings demonstrate that hyperparameter optimization has a moderate impact in simple environments, resulting in a 3.9% and a 12.4% increase in the mean episodic rewards of Basic and Defend the Center scenarios, respectively, but shows significant gains in complex scenarios, achieving a 523.7% and a 203.7% improvement in Health Gathering and Deadly Corridor (skill level 5) scenarios, respectively. These findings provide insights into how DRL agents can be trained using the PPO algorithm in complex environments with multiple challenging tasks through appropriate hyperparameter optimization.

***Correspondence author email address:** areej.fatemah@faculty.muet.edu.pk
DOI: 10.21015/vtse.v14i1.2348

1 Introduction

Reinforcement Learning (RL) is the subfield of Machine Learning (ML) in which agents learn through interactions with the environment and take actions that maximize reward [1]. The RL agent does not initially know what actions provide better rewards; instead, it explores and discovers them through a learning process of trial and error [2]. The state of the environment is observed by the agent, and actions are performed based on the received state. Due to the performed action, the agent receives positive or negative rewards and a new state of the environment. This process is named the Markov Decision Process (MDP) [3].

The agent ultimately develops a policy, which maps the states perceived by the agent to actions and defines how the agent behaves in the environment. Policies can

be deterministic or stochastic, providing the probabilities of taking an action in a specific state. A value function estimates the expected total reward obtained by the agent when the agent is starting from a specific state and then follows a given policy.

Deep Reinforcement Learning (DRL) is the combination of both Deep Learning (DL) and RL, which means Neural Networks (NN) are used to develop the policy and a value function of the RL agent. DRL allows the agents to process raw pixels as states and handle high-dimensional state and action spaces. RL is divided into two categories: the first is Model-free, and the second is Model-based. The model-free algorithms learn with the help of trial and error through interaction with the environment; in model-based algorithms, the model is created for the environment and the possible actions.

Further division of Model-free RL algorithms also happens, as it is also split into two: the value-based and the policy-based. The value-based methods find the estimated value of the given state. Policy-based algorithms directly learn a policy without explicitly estimating a value function. Value-based methods involve algorithms such as Q-Learning, which provides the value of taking an action in a given state through the Q-table [4]. While it is effective for discrete state spaces, it fails in high-dimensional environments.

Deep Q-Network (DQN) is a popular variant of the Q-learning algorithm [5]. It is a type of value-based algorithm that makes use of a Convolutional Neural Network (CNN) to estimate the Q-function. It takes raw pixels as inputs and provides a value estimate. However, DQN fails in environments that have high-dimensional action spaces. With the help of Policy Gradients, the policy-based methods optimize the policy function through the gradient ascent on the expected total reward. Policy Gradients are effective in environments with continuous and high-dimensional action spaces. However, when it comes to early algorithms, such as REINFORCE [6], these suffer mainly from sample inefficiency and high variance, which causes unstable learning. The Advantage Actor-Critic (A2C) is considered a popular algorithm that, with the help of the advantage function, tries to estimate the goodness of the action as compared to the average actions taken in the given state [7, 8]. The Asynchronous Advantage Actor Critic (A3C) is an alternative that allows training multiple agents independently in parallel environments. Although with policy gradients, sampling efficiency and step size in policy updates still faced problems. The Trust Region Policy Optimization (TRPO) provided the solution to this problem by limiting the policy updates to a trust region, but it was computationally complex [9].

The alternative to this is Proximal Policy Optimization (PPO), which is much simpler and efficient [10]. PPO introduced a clipped surrogate objective function that restricts large policy updates without requiring complex second-order Optimization. Due to the ease of implementation and efficient performance in a wide range of environments, the PPO algorithm is widely considered in modern research.

Video games are an ideal avenue for evaluating these RL algorithms due to the diverse environments and complex challenges that require RL agents to develop quick planning and real-time decision-making skills. Doom is a first-person shooter game requiring efficient navigation, combat, and resource management skills. ViZDoom is an established platform used by researchers in DRL [11, 12]. Based on the old Doom first-person shooter game engine, the most appealing factor about ViZDoom is the balance between computational resources and 3D visuals. What makes ViZDoom different from most heavy simulators is its ability to render a thousand frames per second on one CPU core, making it well-suited for RL experiments [13].

The foundational research conducted by Kempka et al. introduces ViZDoom as a research platform designed for visual RL using the classic Doom game [14]. The authors demonstrated the platform's capabilities by successfully training bots from visual input using convolutional deep neural networks and applying deep Q-learning in various Doom scenarios. Their experiments compared agent learning progress and final performance in both a basic scenario and a health gathering task. The agents were evaluated based on the average achieved score in the scenarios. With this study, ViZDoom has proven to be a powerful platform for visual-based RL research.

ViZDoom has since been widely used to evaluate RL agents in Doom-based environments. Wydmuch et al. present the first two editions of the Visual Doom

AI competition held in 2016 and 2017 [12]. It focused on bots competing in a multi-player Deathmatch scenario using only visual information. The winning bots employed modern algorithms such as A3C, DRQN, and DFP. Although the RL-trained bots showed great competence, they were still outperformed by human players. Karttunen et al. address the sim-to-real gap in RL, focusing on how policies learned in a virtual environment using ViZDoom can be transferred to a real-world robot (Turtlebot 3 Waffle) despite differences in action spaces between simulation and the real world [15].

The study utilized DQN and PPO algorithms and enabled agents to adapt to new or limited action sets in the real world. The results indicated that with relatively little additional training, agents trained in ViZDoom could generalize and operate effectively in real-world tasks, showcasing the robustness of the approach even when action sets are incomplete or non-identical. Zakharenkov and Makarov conducted a comparative study of DQN and PPO within the challenging ViZDoom Deathmatch scenario [16].

Both the DQN and PPO agents were trained with and without a pre-trained object detection neural network providing object coordinates as input. The game metrics, such as kills, deaths, medkits collected, and the kill/death ratio, were used as the criteria for evaluation. Their findings revealed that DQN performs better in navigation-related tasks, while PPO offered more stable learning in other complex related settings, demonstrating the trade-offs between value-based and policy-gradient methods in RL.

While previous studies have investigated several RL algorithms, including the DQN, A3C, and PPO algorithms across ViZDoom scenarios ranging from basic navigation to more challenging environments involving combat, little attention has been given to the impact of systematic hyperparameter tuning on the performance of the PPO algorithm implemented in ViZDoom and the application of advanced RL techniques such as reward shaping and curriculum learning, specifically in the Deadly Corridor scenario [16–18].

To address this research gap, this study aims to train RL agents using the PPO algorithm with the help of the Stable-Baselines3 (SB3) framework across four ViZDoom scenarios: Basic, Defend the Center, Health Gathering, and Deadly Corridor. The research applies reward shaping and curriculum learning in the Deadly Corridor scenario and performs hyperparameter tuning in each scenario using Optuna. Optuna is a powerful adaptive framework that is highly effective for hyperparameter optimization in software intensive environments as it provides a balance between exploration vs exploitation [19].

It analyzes the impact of hyperparameter optimization on the PPO agent's learning speed and overall performance to construct a deep understanding behavior of RL agents trained with different hyperparameters using the PPO algorithm in challenging game environments.

The remainder of this paper is structured as follows: we present the hyperparameter tuning approaches next, followed by the environmental setup and RL pipeline. The experimental configurations are then detailed, followed by the training results and a discussion on the impact of hyperparameter tuning. Lastly, we present challenges and limitations of the research followed by the conclusion.

2 Hyperparameter Tuning Approaches

Choosing Hyperparameters is an important part of training an RL agent, as the achieved performance by the RL Agent is directly related to the hyperparameters used to train it. With a high learning rate, the training time becomes short, and the performance of the agent may be negatively impacted. If the learning rate is lower, the training time of the agent increases [20]. Similarly, the influence of hyperparameter tuning is greater in the RL domain, where computationally expensive training is involved, and improper hyperparameters, such as modifications to the learning rate, discount factor, clipping threshold, or number of steps, would downstream the agent's decision-making performance in the environment.

Grid search and random search are the traditional approaches that have been used for a long time in machine learning. To fine-tune the hyperparameters using the grid search approach, it searches sequentially for every possible combination of specified values of hyperparameters, but there is a downside to this approach: the grid search becomes a computationally

expensive process and an unusable approach when the number of hyperparameters that need to be fine-tuned increases. Another approach, Random search, tunes hyperparameters by taking random samples of hyperparameters.

The performance of random search is better than the grid search, as it tries not to search the unproductive region of the search space. Both approaches, however, are not adaptive as they do not learn from past evaluations, which tends to cause inefficiencies, particularly for complex, large models. The problem is overcome by using one of the advanced approaches, like Bayesian Optimization, which develops surrogate models of the objective function and applies acquisition functions to find the stability between exploration and exploitation of the environment [20, 21].

A modern approach to tackling the problem of choosing fine-tuned hyperparameters is Optuna. Optuna is one of the best open-source systems that uses modern samplers like the Tree-Structured Parzen Estimator (TPE) sampler. TPE learns from its past trials, unlike grid and random search, to inform the search towards more fruitful hyperparameter areas. One of the main advantages of using Optuna for Hyperparameter tuning is its pruning mechanism, which enables the tuning process to stop trials early that are low-performing. This saves time significantly in RL, which involves hours of training trials. As the underperforming trials are stopped, it also saves the computational resources while tuning the hyperparameters [14].

3 Environmental Setup and RL Pipeline

The research workflow for RL with PPO on VizDoom has been outlined in Figure 1.

3.1 ViZDoom

Fundamentally, ViZDoom gives the raw pixels as the state of the environment, very closely approximating how humans perceive and respond in video games. This makes the environment challenging; the agent not only needs to identify objects like weapons, health kits, or enemies, but also needs to make decisions and perform actions under sparse and delayed rewards.

One of the most beneficial factors of ViZDoom is its flexibility in scenario creation. Researchers can use the default environments provided by ViZDoom, such as Basic, Defend the center, Health Gathering, and Deadly Corridor. It also allows using third-party libraries on top of ViZDoom for various environments and tools to help in the process of RL research. ViZDoom also provides the ability to create custom environments using different map editors.

Basic Scenario

This scenario is used to check if it is practical to train an agent in a 3D environment. The map is designed in the shape of a rectangle. The player is spawned in the middle. The main objective of the player in this scenario is to kill the monster, which appears along the opposite side of the wall. To complete the objective, the player can move left, right, or shoot the monster. The monster can be killed with just one hit from the player. The conditions for an episode to come to an end are to eliminate the target enemy or when the episode times out.

Reward

The reward structure of the basic scenario teaches the RL agent some basic combat and survival skills.

- When the agent kills the monster, it receives a large reward of +106. Since this is the primary objective, the system makes it obvious that the most beneficial course of action is to destroy the enemy.
- The agent is penalized negatively with a -5 reward for each fire. The agent is deterred from wasting ammunition or shooting aimlessly by this penalty.
- The agent is also rewarded negatively with -1 point for every second it stays alive. This negative reward makes the agent kill the enemy as fast as possible.

Configuration

The scenario is purposefully kept extremely basic, providing the agent with only the most necessary configurations.

- It can only move left, right, or shoot. This maintains the learning process's emphasis on the fundamental abilities of timing and aim.
- The fact that the ammo count is the only game variable that must be kept in mind emphasizes how important every shot is.
- Each episode has a time limit of 300 timesteps to keep the game from going on forever. The episode

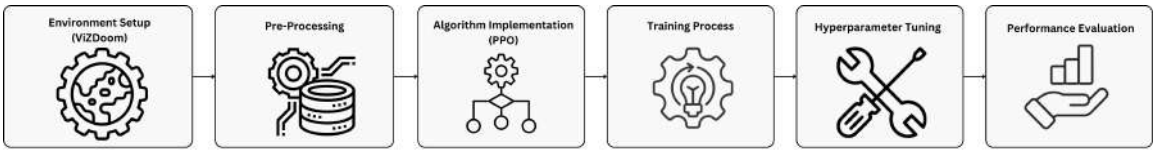

Figure 1. Research Workflow

simply ends if the monster is not killed in that amount of time.

Defend The Center Scenario

The purpose of this scenario is to teach the RL agent that eliminating the monsters is a GOOD thing, which means the RL agent will be rewarded positively. On the other hand, when the monsters eliminate the player, it indicates a BAD thing, which means the agent will be rewarded negatively. In addition to those basic rules, wasting a large number of bullets will not be counted as a good thing. The RL agent is rewarded positively only for eliminating the monsters around him, so it has to figure out the rest for itself. The map of this scenario is a large circle. The scenario starts by fixing the player at the center of the circle. The monsters are generated along the wall of the circle, and the elimination of the monsters can be done with a single shot. After dying, each monster is regenerated after a certain time. The episode only ends with the elimination of the player. It is inevitable due to limited ammo.

Reward

The reward structure of the Defend The Center scenario is to teach the RL agent how to defend itself in the circle.

- When the player eliminates the monster, the player will be rewarded with the +1. Since this is the primary objective of the player, the reward is positive.
- On the other hand, when the monsters eliminate the player. The player is penalized with the negative reward, which is -1.

Configuration

- There are only 3 available actions that a player can take in order to defend itself.
  - The player can turn left.
  - The player can turn right.
  - The player can shoot to eliminate the monsters.
- There are 2 game variables configured in this scenario:
  - **"Player's health,"** which determines the capacity of the player to take damage.
  - **"Ammo,"** which is the amount of bullets the player can use to eliminate the monsters.
- The timeout of the episode for this scenario is set to 2100 timesteps.
- The setting of the difficulty level of the scenario by default is 3.

Health Gathering Scenario

The main aim of this scenario is to teach the RL agent how to survive without knowing what makes the agent survive. Here, the RL agent only gets to know the value of life and learns that death is bad. As the agent knows the value of life, the agent tries to prolong his life span and learns that his health is connected with it. In this scenario, the map is rectangular in shape with a green, acidic floor, which reduces the health of the player with the passage of time. At the start, some medkits are uniformly spread over the floor. At random time intervals, new medkits appear from the sky. The purpose of the medkit is to increase some portion of the player's health, which means that to prolong the agent's life, it needs to pick them up. There are only two conditions to finish the episode of this scenario: the player's death or the episode timeout.

Reward

The reward structure of this scenario is to teach the agent how to prolong its life span.

- The agent is rewarded positively with +1 point for surviving each timestep of the game. This makes the RL agent prioritize the survival time of the episode.

- On the death of the agent, he is penalized with an extreme negative reward, which is -100 points. This extreme negative reward makes the agent try harder to avoid death.

Configuration

- There are 3 available actions that an agent can take in order to make his life span longer. The agent can turn left, right, and move forward.
- The player's health is the only game variable that is used in this scenario.
- The episode does not have a fixed timeout, and it only ends when the player dies.

Deadly Corridor Scenario

This scenario teaches an agent to navigate effectively to obtain its primary objective, which is the vest. While in the process of obtaining it, the player has to ensure its survival as the enemies try to kill the player. The environment of this scenario consists of a corridor with shooting monsters on either side. Six monsters in total are placed in the corridor. A green vest is situated at the terminal end of the corridor. The reward system of the scenario is based on the distance between the RL agent and the vest. It may be positive or negative. With the increase in the distance, it is negative, and it is positive with the decrease in the distance. If the agent neglects the monsters positioned along the sides and moves directly towards the vest, the player will be killed somewhere along the way. To make the scenario more challenging for the agent, the difficulty level is 5. Since this scenario is more complex than the other scenarios, the curriculum learning and reward shaping techniques have been used to help the agent achieve its objective efficiently.

Reward

In addition to the default rewards, reward shaping has been applied to provide auxiliary rewards to the agent, which are as follows:

- The agent is rewarded with +dX points for moving closer to the vest. The agent is rewarded with -dX points when moving away from the vest. If the agent dies before obtaining the vest, the agent is penalized with extremely negative points of -100.
- A negative reward is assigned when the player takes damage during the episode, which is then multiplied by 10 and added to the complete reward function. This emphasizes to the agent that it should avoid taking damage.
- A positive reward is given for the number of monsters that are hit during the episode, which is then multiplied by 200 and added to the complete reward function. This pushes the agent to kill the monster for a greater reward.
- A negative reward is issued when the player fires a shot, and the ammo is decreased, which is then multiplied by 5 and added to the complete reward function. This tells the agent to use the ammo wisely.

Configuration

- The episode timeout for this scenario is set to 2100 timesteps.
- There are seven available actions the player can perform in order to achieve the objective in this scenario.
- Move forward, Move backward, Move left, Move right, Turn left, Turn right, Attack (shoot).
- The difficulty level of the scenario is 5 (maximum) by default. Although in order to apply the curriculum learning, the scenario difficulty level is placed between 1 to 5.
- The player's health variable is the only game variable used in this scenario by default.
- The following are the game variables used to achieve curriculum learning: Ammo, Hitcount, and Damage taken.

3.2 ViZDoom Environment Setup and Configuration

The following are the tools and libraries used to set up and configure the RL environment.

- **Stable-Baseline3**, a PyTorch upgraded version of stable-baselines2. This is a library offering a modern implementation of various popular RL algorithms. It provides the implementation of the PPO algorithm used by the RL agent for training in the RL environment.
- **ViZDoom,** a research platform, which provides the Doom game environment. It enables the RL agent to have game state access and perform actions pro-

grammatically.

- **Gymnasium** an API (application programming interface) for all single-agent RL environments, with implementations of common environments: cart-pole, pendulum, mountain-car, mujoco, atari, and more [21]. It ensures that different environment follows a consistent structure, so it will be way easier to train an RL agent with an existing RL algorithm.
- **OpenCV,** a computer vision library. It is used to pre-process the game state perceived by the RL agent, so that it saves unnecessary computational efforts.

Initialize Environment

A crucial step to configure the RL environment is to initialize the ViZDoom environment using the required scenario configuration file, so that the environment can be created according to the required configuration and scenario. The initialization of the ViZDoom environment pushes it to the next step, where the RL agent can perform actions in the ViZDoom environment and receive rewards to shape its policies.

3.3 Pre-processing Pipeline

The main focus while dealing with the RL agent is to improve the performance of the agent, which is a basic requirement. Often, less attention is given to the stage of data pre-processing, as the training of the RL agent by providing the visual frames as input is a computationally expensive process. For the purpose of reducing the complexity of data from the ViZDoom environment, the preprocessing of data is needed. Through the data preprocessing pipeline, the time required to train an RL agent and the memory used by the RL agent during training can be reduced. To achieve the benefits of data pre-processing, the project implements the following operations in the pipeline of the visual data from the ViZDoom environment before passing it to the RL agent:

- Channel Rearrangement.
- Grayscale Conversion.
- Resizing.
- Reshaping.

3.3.1 Channel Rearrangement

In this phase, the channel of the visual frame coming from the ViZDoom environment is rearranged. The default channel arrangement of the visual frame is (Channel, Height, Width).

- The channel axis provides information regarding the color of the visual input.
- The Height gives information about the height of the visual input.
- The Weight axis gives information about the width of the visual input.

The purpose is to change the positions of the axis of the visual input frame from (Channel, Height, Width) to (Height, Width, Channel).

3.3.2 Grayscale Conversion

In grayscale conversion, OpenCV is used to convert the RGB color channel of the visual frame to a grayscale color channel. There are three color channels of RGB, which are then changed into one color channel, which is then called a grayscale color channel. The input shape of this operation is (Height, Width, Channel), and it is then converted as an output to (Height, Width).

3.3.3 Resizing

This phase takes the input, which was generated as an output of the grayscale conversion operation. In this phase, the resizing of the image occurs, and the resolution of the image is downscaled. The exact resized image is 100 x 160. 100 pixels is the height, and 160 pixels is the width. The purpose of downscaling the image is to reduce the number of pixels in the image to be processed by the RL agent, which helps to decrease the computational cost of processing the image.

3.3.4 Reshaping

Reshaping is the last operation, which is performed on the visual image after resizing in the pre-processing pipeline. It takes a resized image as input. In this phase, the 2D grayscale image (100,160) is reshaped again to a 3D image by converting to the channel-last format. That is (100, 160, 1). The main purpose of this phase is to make the image format acceptable in the gymnasium environment.

3.3.5 Reward Shaping

In RL, a commonly faced issue is of rewards being sparse. It means that there are very few states and actions that

will give the action feedback to non-zero rewards. This creates problems as initially, RL algorithms behave completely randomly. Due to this, they will have a problem finding good rewards. This problem can be overcome with reward shaping. Reward shaping or designing is used as a technique that helps the RL agent learn actions efficiently during the learning process. The reward shaping is the process of modifying the structure of rewards provided by the environment as a result of taking actions by the RL agent. Reward shaping in RL helps to provide extra response information that provides assistance to the RL agent to evaluate its action and accelerates its learning process.

The implementation of reward shaping has been done in the deadly corridor scenario of ViZDoom. The calculation of the total reward gained by an RL agent for each step is done with the movement reward provided by the environment and extra reward information provided by the game variables. The reward function is adapted based on an implementation in prior work. The following are the game variables used for reward shaping in the deadly corridor scenario:

- **DAMAGE TAKEN:** This variable of VizDoom provides information about the amount of damage taken by the player during the present episode of gameplay.
- **HITCOUNT:** In a VizDoom environment, the number of hit monsters, players, and bots in the present episode is calculated by the HitCount variable.
- **SELECTED WEAPON AMMO:** This variable provides information regarding the ammo of the selected weapon.

Components of the Shaped Reward

The following are the components of the shaped reward that help to calculate the total reward for each step:

- **Movement Reward:** This is the reward that the RL agent gets directly returned from the environment, which is proportional to the change in distance between the player and the vest. The movement reward is positive with the action taken by the RL agent, which brings it closer to the vest; on the other hand, the movement reward is negative with an increase in the distance between the RL agent and the vest.
- **Damage Taken Change:** A negative reward is applied if the player takes damage from monsters. To calculate the change of damage taken by the RL agent, previous damage taken is subtracted from the current damage taken to calculate the reward for the RL agent. It is then multiplied by 10 and added to the total reward.
- **Hitcount Change:** There is a positive reward for the RL agent when it hits the enemy. To calculate the reward for hitting the enemy, the hit count change is calculated through the difference between the current and the previous hit count of the timestep. The increase in the hit count is multiplied by 200, so that the RL agent focuses more on hitting an enemy.
- **Ammo Change:** A small penalty is applied when ammo decreases. The change in ammo of the RL agent is calculated through the difference between the previous and the current ammo of the timestep. It is multiplied by 5 and added to the total reward.

Curriculum Learning

In RL, most of the time, the task that the agent has to learn is hard, and to successfully complete that task, the agent needs to acquire the incremental acquisition of skills; for instance, when one wants to make a bipedal agent learn to go through hard obstacles, it must first learn to stand, then walk, then maybe jump. There are variations in the environment (that affect the difficulty), and one wants the agent to be robust to them. In such cases, it seems necessary to propose different-level tasks to the RL agent and organize them such that the agent progressively acquires skills. This approach is called Curriculum Learning and usually implies a hand-designed curriculum (or set of tasks organized in a specific order) [23].

Deadly Corridor Levels 1–5

The implementation of the curriculum learning has been done in the deadly corridor scenario of ViZDoom, in which the agent is trained with a sequence of progressively more difficult levels. The deadly corridor scenario difficulty level is not fixed by the VizDoom environment; rather, it provides the flexibility to set the difficulty level according to the training needs. The difficulty level of the scenario can be set between 1 to 5. The curriculum

learning has been implemented by using these difficulty levels. It begins by setting the scenario difficulty level to 1. This is the basic difficulty level of the scenario that helps the RL agent to explore the environment. The training of the RL agent begins with difficulty level 1 of the deadly corridor scenario and gradually increases the difficulty to the maximum level of 5.

Progression Criteria

In curriculum learning, the RL agent is trained on progressively harder tasks, so the difficulty level gradually increases. To decide whether the RL agent is trained enough to increase its difficulty level, there must be a progression criterion, which helps to measure the skills that are acquired by the RL agent. Progression criteria decide when the RL agent is ready to move from one stage to the next stage with an increased difficulty level. The progression criteria for this curriculum learning are set to timestep-based, which means a fixed number of time steps is set for each difficulty level of the scenario. The RL agent was trained for a fixed number of time steps in difficulty level 1 and then moved to the next difficulty level of the scenario. The following are the criteria of progress that are set for each difficulty level:

- Because of allowing the agent to explore the environment, 260,000 steps is set as a criterion for difficulty level 1.
- 40,000 timesteps are set for level 2.
- The same number of steps is set for level 3, which is 40,000.
- The timesteps are set to 40,000, also for the difficulty level 4.
- For difficulty level 5, the number of steps is set to 170,000 to achieve optimal performance in this difficulty.

Training Transfer Between Levels

With standard RL settings, at the beginning of the training, the agent takes actions using a random policy and directly tries to learn the optimal policy. When the assigned task is difficult, the learning of the RL agent is very slow [22]. Transfer learning is a great method that provides a way to reduce the time of training of an RL agent. The main purpose of transfer learning is that, rather than directly learning the target difficulty level 5 of the deadly corridor scenario, the agent can first train on the scenario with difficulty level 1 and transfer the knowledge that has been learned by the agent through the training to solve the next difficulty level of the ViZDoom scenario. In this research, the knowledge is transferred in the form of a saved model across the difficulty levels of the scenario. This means the model trained for difficulty level 1 of the scenario is used to initialize the model for training the next difficulty level of the scenario.

3.4 Proximal Policy Optimization Algorithm

The PPO algorithm is used for training the RL agent [11]. Stable Baseline3 (SB3) is the RL library, which is used in the application of the PPO algorithm in training [24]. PPO uses Policy Gradient, which is used to improve the policy of the RL agent directly by optimizing its policy parameters. It improves the learning of the agent and stabilizes it by making sure that the changes made to the policy during the training of the RL agent are not massive. This ensures that the agent makes updates to the policy that are within a trust region and does not experience unstable training in the environment. PPO uses a technique called clipping directly in the objective function, which is called the clipped surrogate objective function, which clips the ratio between the current policy and the old policy of the agent to a range between $1 - \epsilon$ and $1 + \epsilon$. This gives the measurement of the change in the RL agent policy, which shows how much the change occurs between the current policy and the previous policy. All this is done to make sure that the change gap between the current policy and the previous policy does not become large.

The ratio between the current policy and the previous policy is calculated as:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{1}$$

Where;

- $r_t(\theta)$ is the probability ratio at timestep t.
- $\pi_\theta(a_t \mid s_t)$ is the probability of the current policy for taking the action at in the state st at timestep t.
- $\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ is the probability of the old policy for taking the action at in the state st at timestep t

If $r_t(\theta) = 1$, both policies assign the same probability for taking the action a_t in the state s_t at timestep t. If $r_t(\theta) > 1$,

the probability of taking the action at in the state st at timestep t will be higher in the current policy than the old policy. Whereas, if $r_t(\theta) < 1$, the probability of taking the action at in the state st at timestep t will be lower in the current policy than in the old policy.
The clipped surrogate objective function is given as follows:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \quad (2)$$

Where;

- $L^{\text{CLIP}}(\theta)$ is the loss function for the actor network with the θ parameters.
- $\hat{\mathbb{E}}_t$ is the expected value over t timesteps.
- $r_t(\theta)$ is the probability ratio between the current policy and the old policy at timestep t.
- $\hat{A}_t$ is the advantage estimate at time t. It estimates that the action taken by the agent is either better or worse compared to the average action in the state s.
- ϵ is the clipping parameter that defines the range of the allowed ratio between the two policies. Its value is usually 0.2 or 0.1.
- clip($r_t(\theta)$, $1-\epsilon$, $1+\epsilon$) clips the ratio to stay in the range between $1-\epsilon$ and $1+\epsilon$

The clipped surrogate objective function includes a non-clipped probability ratio and a probability ratio that is clipped between the ranges $1-\epsilon$ and $1+\epsilon$. The probability ratios are then multiplied by the Advantage estimate. The function then takes the minimum between the clipped and the non-clipped objective based on the probability ratio and the advantage. This operation is done to make sure that the gap between the current policy and the previous policy does not become significant, and ensures that the updates that have been made to the policy are small, which contributes to making the agent's policy stable. The PPO algorithm also uses a value function for the critic network that calculates the expected return for taking an action by the RL agent in a given state:

$$L^{VF}(\theta) = \mathbb{E}_t \left[\left(V_\theta(s_t) - V_t^{\text{target}} \right)^2 \right] \quad (3)$$

Where;

- $L^{VF}(\theta)$ is the loss function for the critic network with the θ parameters.
- $V_\theta(s_t)$ is the predicted value for the state s_t.
- V_t^{target} is the expected discounted return from the state s_t

An entropy bonus is added to PPO, which fosters randomness in the agent and encourages exploration:

$$S[\pi_\theta](s_t) = \mathbb{E}_{a\sim\pi_\theta} \left[-\log \pi_\theta(a \mid s_t) \right] \quad (4)$$

Where $S[\pi_\theta](s_t)$ is the entropy of the policy distribution at state s_t.
A higher entropy means a more uniform distribution, which encourages the agent to explore, whereas a lower entropy forces the agent towards exploitation. The overall PPO Loss function involves the clipped surrogate objective function, the value function loss, and the entropy bonus term. The complete PPO loss function is given as:

$$L_t^{\text{CLIP+VF+S}}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right] \quad (5)$$

Where:

- c_1 is the coefficient of the value function loss. Its value is usually 1.0
- c_2 is the coefficient of the entropy. It's usually kept at 0.01

3.5 Training Process

The training of the RL agent is carried out by using ViZDoom, Gymnasium, and SB3. The training process enables the agent to make strategic and optimal decisions that are important for achieving scenario objectives. The custom environment class by Gymnasium allows for easy integration with other RL libraries and tools such as SB3. The custom environment class is used for preprocessing, which is detailed in Section 3.3. The PPO algorithm is implemented using SB3, along with custom callbacks for saving the models. The **CnnPolicy**, provided by SB3, is used for processing images, since the observations received by the agent from the environment are images in the form of raw pixels. Here, the 'CnnPolicy' is an SB3 class that handles all the networks. The architecture of the CnnPolicy is comprised of two separate components:

- A feature extractor, which is a CNN that extracts features from images.
- A fully-connected network that maps the extracted features to the actions/values.

The features of the visual frame are extracted by using the NatureCNN architecture along with a linear layer after the CNN [9]. The actor network and the critic network share the CNN in the PPO algorithm implementation [24]. A custom callback is created using SB3 that saves the model's parameters periodically after every 10,000 steps for later evaluation. The models that were saved can be loaded again later, so that the trained agent's performance can be evaluated and compared. This helps in continuing the training of the RL agent for further improvement. Moreover, SB3 saves the logs during the training of the model periodically after every N steps, which include: episode length mean, episode reward mean, total timesteps the model has been trained, time elapsed since training of the agent, loss values, and other training statistics. The logs can be visualized using TensorBoard [25], which allows monitoring and analyzing the agent's performance throughout the training.

In the beginning, RL agents are trained using the default parameters for PPO provided by SB3 in the basic, defend the center, health gathering, and deadly corridor scenarios. The parameters provided by default by SB3, used to initially train the RL agents are given in Table 1:

Table 1. Default Hyperparameters for PPO Algorithm

Parameter	Value
Learning rate	0.0003
N steps	2048
Batch size	64
N epochs	10
Gamma (γ)	0.99
GAE Lambda (λ)	0.95
Clip range (ϵ)	0.2
Max grad norm	0.5

Where:

- Learning rate controls the size of updates made to the policy.
- N steps is the number of steps collected per policy update.
- Batch size is the size of the mini-batch.
- N epochs is the number of epochs when the surrogate loss is optimizing.
- Gamma (γ) is the discount factor.
- GAE Lambda (λ) is a parameter for Generalized Advantage Estimation that balances bias and variance.
- Clip range (ϵ) is the clipping parameter ϵ employed in the PPO's clipped surrogate objective function.
- Max grad norm is the maximum value for gradient clipping.

For each scenario, the RL agent is trained for a fixed timestep. Simple environments, like the Basic scenario, require the agent to be trained on fewer timesteps to learn an optimal policy, whereas environments that are more complex, like the Deadly Corridor scenario, require more timesteps for the agent to learn an optimal behaviour in those environments. The total number of timesteps the RL agents are initially trained for in each scenario is provided in Table 2.

Table 2. Training Timesteps for Each Scenario

Scenario	Timesteps
Basic	50,000
Defend the center	80,000
Health Gathering	80,000
Deadly Corridor	100,000

The results of training the agent initially on the default set of hyperparameters (Table 1) for the number of timesteps given in Table 2 are provided in Section 5.

3.6 Hyperparameter Tuning

Basic Scenario Hyperparameters

Since the Basic scenario is quite simple and does not involve any complexities, the hyperparameter optimization process focuses on two parameters that have a significant impact on the performance of the RL agent: the learning rate and gamma. The search space defined for the hyperparameters for the Basic scenario is presented in Table 3.

Table 3. Basic Scenario Hyperparameter Search Space

Parameter	Search Space Range	Type	Distribution
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-2}]$	Float	Log-uniform
Gamma (γ)	[0.9, 0.999]	Float	Log-uniform

The trials were conducted for a fixed number of 50,000 timesteps to allow the agent to try different actions and explore the environment. During the process

of optimizing the hyperparameters, the mean reward of the episodes is not the same for each trial; its value ranges from -300 to 82.3. The optimal hyperparameters for the Basic scenario are presented in Table 4.

Table 4. Basic Scenario Optimal Hyperparameters

Parameter	Value
Learning rate	0.0002
Gamma (γ)	0.988

Defend The Center Scenario Hyperparameters

The Defend the center scenario is more complex than the basic scenario, encompassing a bigger environment, an increased number of actions the agent can perform, and a higher count of enemies that the agent can kill; therefore, the hyperparameters that are chosen to be optimized for this scenario are the n steps, learning rate, clip range (ϵ), gamma (γ), and GAE Lambda (λ). The search space defined for the hyperparameters for the Deadly Corridor scenario is presented in Table 5.

Table 5. Defend The Center Scenario Hyperparameter Search Space

Parameter	Search Space Range	Type	Distribution
N steps	[2048, 4096]	Integer	Categorical
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-1}]$	Float	Log-uniform
Gamma (γ)	[0.9, 0.999]	Float	Log-uniform
Clip Range (ϵ)	[0.1, 0.2]	Float	Uniform
GAE Lambda (λ)	[0.8, 0.99]	Float	Uniform

The trials were carried out for a fixed number of 80,000 timesteps. The process of optimizing the hyperparameters produces different values of the mean rewards of the episodes for each trial, and these values range from -1 to 13.6. The optimal hyperparameters for the Defend The Center scenario are presented in Table 6.

Health Gathering Scenario Hyperparameters

The Health Gathering scenario requires the agent to collect health packs in order to survive longer on an acidic floor that decreases the health of the agent with the passage of time, which increases the complexity of the environment. These are the hyperparameters that are optimized by hyperparameter tuning for this scenario of ViZDoom: n steps, learning rate, clip range, gamma (γ), and GAE Lambda (λ). The search space defined for the hyperparameters for the Health Gathering scenario is presented in Table 7.

Table 6. Defend The Center Scenario Optimal Hyperparameters

Parameter	Value
N steps	4096
Learning rate	0.0001
Gamma (γ)	0.93
Clip range (ϵ)	0.12
GAE Lambda (λ)	0.96

Table 7. Health Gathering Scenario Hyperparameter Search Space

Parameter	Search Space Range	Type	Distribution
N steps	[2048, 4096, 8192]	Integer	Categorical
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-4}]$	Float	Log-uniform
Gamma (γ)	[0.8, 0.999]	Float	Log-uniform
Clip Range (ϵ)	[0.1, 0.2]	Float	Uniform
GAE Lambda (λ)	[0.8, 0.99]	Float	Uniform

For each trial, the RL agent was trained for a fixed number of 80,000 timesteps. The mean episodic reward was different, ranging from 284 to 1831, for each trial during the optimization process. The optimal hyperparameters for the Health Gathering scenario are presented in Table 8.

Table 8. Health Gathering Scenario Optimal Hyperparameters

Parameter	Value
N steps	8192
Learning rate	0.0001
Gamma (γ)	0.99
Clip range (ϵ)	0.2
GAE Lambda (λ)	0.95

Deadly Corridor Scenario Hyperparameters

The complexity of this scenario is higher than all the other scenarios, since it involves developing a strategy to kill all the enemies that shoot at the agent in the corridor and reach the vest at the end of the corridor

while managing its health and ammunition. In this scenario of ViZDoom, the RL agent learn the skills of navigation, combat, and resource management. The hyperparameters that are chosen to be optimized for this scenario are the n steps, learning rate, clip range, gamma, and GAE Lambda. The search space defined for the hyperparameters for the Deadly Corridor scenario is presented in Table 9.

Table 9. Deadly Corridor Scenario Hyperparameter Search Space

Parameter	Search Space Range	Type	Distribution
N steps	[2048, 4096, 8192]	Integer	Categorical
Learning Rate	$[1 \times 10^{-5}, 1 \times 10^{-3}]$	Float	Log-uniform
Gamma (γ)	[0.9, 0.999]	Float	Log-uniform
Clip Range (ϵ)	[0.1, 0.2]	Float	Uniform
GAE Lambda (λ)	[0.8, 0.99]	Float	Uniform

Due to the complex nature of the environment, the trials for the Defend The Center scenario are conducted on a simplified version of the environment using Doom skill level 1, in which enemies are less deadly and perform less damage to the RL agent, which allows the agent to explore the environment, helping it to learn what actions lead to an optimal policy. Each trial was performed for a fixed number of 100,000 timesteps. However, the optimization process led to the discovery of a critical limitation: none of the 20 trials put together a policy that provided satisfactory results. When the policies learned from the trials were analyzed, they revealed that the RL agent's policy leads to an undesirable behaviour, where the agent runs directly towards the vest while ignoring all the enemies present in the corridor. This strategy works out for the environment where the doom skill level is set to 1, due to the enemies dealing minor damage to the RL agent, but the agent fails catastrophically to achieve the goal when the doom skill level is set to 5, which is the default and recommended doom skill level for the Deadly Corridor scenario. The agent miserably fails to reach the vest at the end of the corridor without eliminating all the enemies first that are shooting at the agent.

The trials identified an unexpected behaviour and maximized the agent's performance through it; however, this undesirable behaviour doesn't help the agent to achieve the task on the required doom skill level. In this scenario, the alternative hyperparameter set was used on the basis of implementation in previous work because of the limitations of the optimization process. The optimal hyperparameters for the Deadly Corridor scenario are presented in Table 10.

Table 10. Deadly Corridor Scenario Optimal Hyperparameters

Parameter	Value
N steps	8192
Learning rate	0.00001
Gamma (γ)	0.95
Clip range (ϵ)	0.1
GAE Lambda (λ)	0.9

Using these hyperparameters, the RL agent was further trained on increasing Doom skill level difficulty from 1 to 5, detailed in Section 3.3.5. The hyperparameters achieved a better performance than the default parameters and the parameters obtained through the optimization process. The agent successfully achieved an optimal behaviour where it learned to eliminate all the enemies in the corridor first and then reach the vest.

4 Experimental Configurations

This section expresses the details of the experimental setup used in the training and assessment of the RL agents, which are developed to take actions in the ViZDoom environments.

4.1 Hardware and Software Configuration

All experimental procedures were executed by utilizing Google Colab, a cloud-oriented Jupyter notebook platform that facilitates access to both complimentary and subscription-based computational resources [26]. In order to fulfill the computational requirements associated with the training of deep reinforcement learning architectures, 100 Compute Units (CUs) were acquired under the Google Colab Pro+ subscription, thereby guaranteeing access to superior-performance GPUs and prolonged runtime thresholds.

4.1.1 Hardware Environment

The specification of the hardware provided through Colab sessions varied slightly depending on resource allocation, but generally included the following criteria:

- **GPU:** NVIDIA Tesla T4
- **CPU:** Intel Xeon (2 virtual CPUs)

- **RAM:** 13GB
- **Runtime Type:** Python 3

For storage, a Google Drive account was connected with the Colab notebook. The above configurations were enough for the training of the RL agent in ViZDoom scenarios.

4.1.2 Software Environment

The software stack used for implementation consisted of the following major tools and frameworks:

- **Operating System:** Ubuntu (Linux-based environment via Colab)
- **Programming Language:** Python 3.12
- **Stable-Baselines3:** 2.7.0
- **PyTorch:** 2.8.0
- **Optune:** 4.5.0
- **Gymnasium:** 1.2.1
- **ViZDoom:** 1.2.4
- **OpenCV:** 4.12.0.88
- **NumPy:** 2.0.2
- **Matplotlib:** 3.10.0
- **TensorBoard:** 2.19.0

4.2 Implementation Details

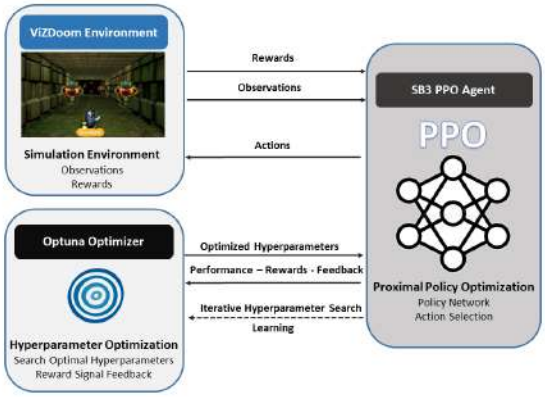

Figure 2. System Architecture

The implementation of every ViZDoom scenario (Basic, Defend the center, Health Gathering, and Deadly Corridor) has been done as a custom gymnasium environment. These custom environments provide seamless integration with Stable-Baseline3. The RL agent used the PPO algorithm for training in the ViZDoom environment. This algorithm is chosen for its efficacy and performance in continuous or high-dimensional action spaces. The overall system architecture for this research is depicted in Figure 2.

4.2.1 Training Parameters

Each scenario was trained with default and optimized hyperparameters determined using Optuna. The models were trained for varying timesteps depending on scenario complexity.

- **Basic:** 50,000 steps
- **Defend the Center:** 80,000 steps
- **Health Gathering:** 80,000 steps
- **Deadly Corridor:** 550,000 steps (with curriculum learning levels 1-5)

The models were saved automatically after each training run for later evaluation and visualization.

4.3 Evaluation Metrics

Evaluating the performance of the generated models is an essential step of all machine learning projects [27]. The evaluation of the trained RL agent has been done using multiple quantitative metrics. These metrics help to estimate the performance and behavior of the RL agent in the environment.

- **Mean Reward:** The average reward obtained by the RL agent is shown by the mean episodic reward. This shows how efficiently the RL agent takes actions to achieve the goal in each scenario.
- **Episode Length:** It shows the average length of an episode served by the RL agent. This helps to estimate how long the agent has survived in the environment.

A qualitative assessment is performed by visually inspecting the behavior of each trained agent to evaluate whether it performs optimally in the given scenario and successfully accomplishes its main objective.

5 Training Results

The results compare and analyze the mean episodic rewards and the mean episode length achieved by evaluating both agents in the four different scenarios.

5.1 Basic Scenario Results

This subsection presents the results for the Basic scenario achieved by training the PPO agent on the default parameters, provided in Table 1, and the optimized parameters, provided in Table 4, for 50,000 timesteps. The results of training the agent on the default parameters and on the optimized parameters are shown in Figure 3 and Figure 4.

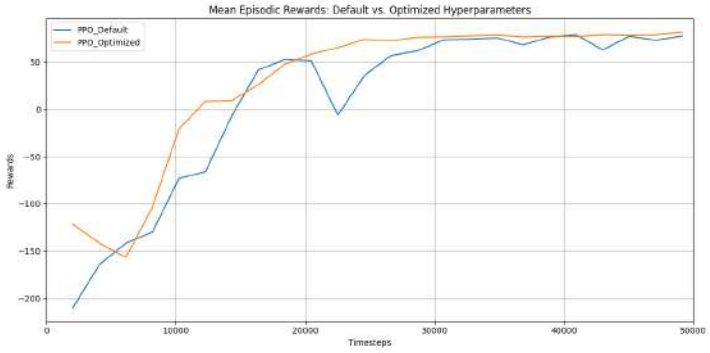

Figure 3. Basic Scenario Mean Episodic Rewards Comparison

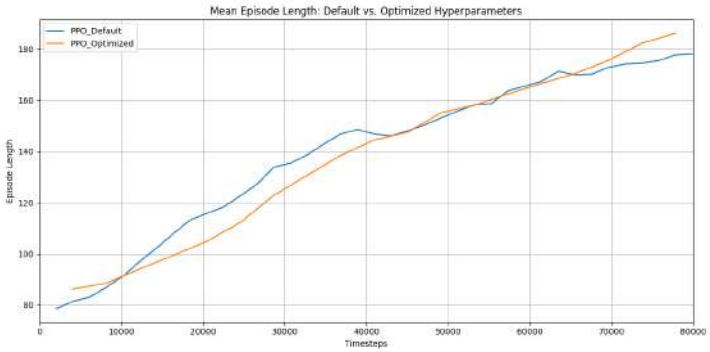

Figure 4. Basic Scenario Mean Episode Length Comparison

Figure 3 compares the Mean Episodic Rewards achieved by the agent trained on the default parameters and the optimized parameters, while the Figure 4 compares the Mean Episode Length for both agents. As the training of the agent progresses, it learns that killing the monster as fast as possible increases the rewards received from the environment, which makes the mean episode length decrease and increases the mean episodic reward. Both the PPO agents are evaluated for 10 episodes in the Basic Scenario using SB3. The PPO agent trained on the default parameters achieves a mean reward of 79.1 with a std reward of 9.08.

Whereas, the PPO agent trained using the optimized parameters achieves a mean reward of 82.2 with a std reward of 8.9. Since the scenario is simple, the agent is able to learn an efficient policy using the default parameters after just 30,000 timesteps, as shown in the Figure 3 and Figure 4.

Therefore, the improvement achieved in terms of the performance of the agent through Hyperparameter optimization is minute, showing that the optimization process has a limited impact in simple environments where the tasks are not highly complex.

5.2 Defend The Center Scenario Results

Here we discuss the results for the Defend The Center scenario achieved by training the PPO agent on the default parameters, provided in Table 1, and the optimized parameters, provided in Table 6, for 80,000 timesteps. The results of training the agent on the default parameters and on the optimized parameters are shown in Figure 5 and Figure 6.

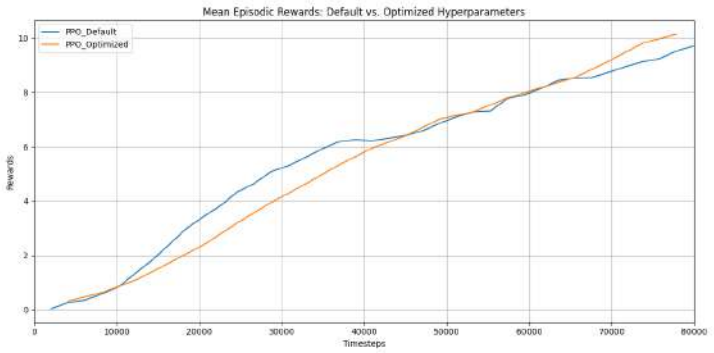

Figure 5. Defend the Center Scenario Mean Episodic Rewards Comparison

As training progresses, the agent learns that killing the monsters provides increased rewards, whereas when the monster kills the agent, it gives negative rewards. It also learns to survive longer by using the ammunition efficiently to kill the monsters. Both the PPO agents are evaluated for 10 episodes in the Defend The Center Scenario using SB3. The PPO agent trained on the default parameters achieves a mean reward of 12.1 with a std reward of 4.5. Whereas, the PPO agent trained using the optimized parameters achieves a mean reward of 13.6 with a std reward of 2.5. Due to the agent's position being fixed in the center and

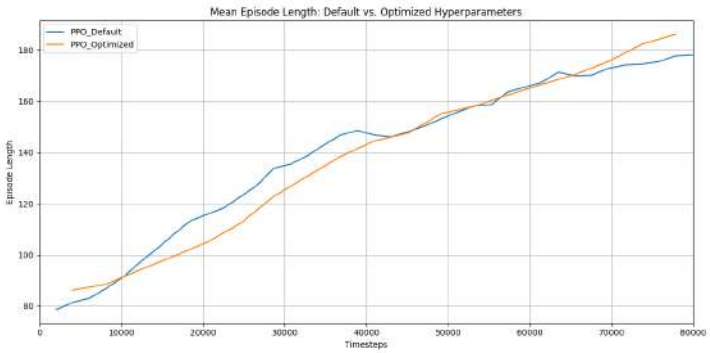

Figure 6. Defend the Center Scenario Mean Episode Length Comparison

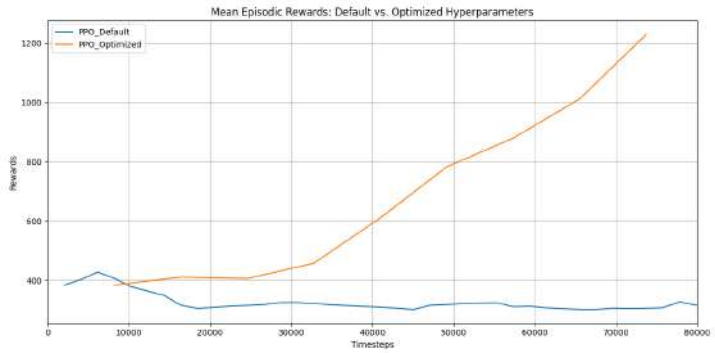

Figure 7. Health Gathering Scenario Mean Episodic Rewards Comparison

the environment being easy, the PPO agent trained on the default parameters learns an efficient policy, as the default parameter values already prove to be well-suited for this scenario. Though the PPO agent trained on the tuned parameters provides an improved performance and displays a stable behaviour throughout the training process, than the PPO agent that is trained on the default parameters.

5.3 Health Gathering Scenario Results

This subsection includes the results for the Health Gathering scenario achieved by training the PPO agent on the default parameters, provided in Table 1, and the optimized parameters, provided in Table 8, for 80,000 timesteps. The objective of the Health Gathering scenario is to teach the agent to survive for a longer time on the acidic floor by collecting the medkits that randomly spawn in the environment. The results of training the agent on the default parameters and on the optimized parameters are shown in Figure 7 and Figure 8.

Both the PPO agents are evaluated for 10 episodes in the Basic Scenario using SB3. By using default parameters for training the PPO agent, its performance achieves a mean reward of 293.6 with a std reward of 28.8. Whereas, the PPO agent trained using the optimized parameters achieves a mean reward of 1831.4 with a std reward of 454.4. In this scenario, the agent trained using the default parameters fails to learn an optimal policy due to the complexity of the environment, as shown in Figure 7 and Figure 8. Whereas the agent trained on the optimized set of parameters learns an optimal policy and survives for a longer time, which

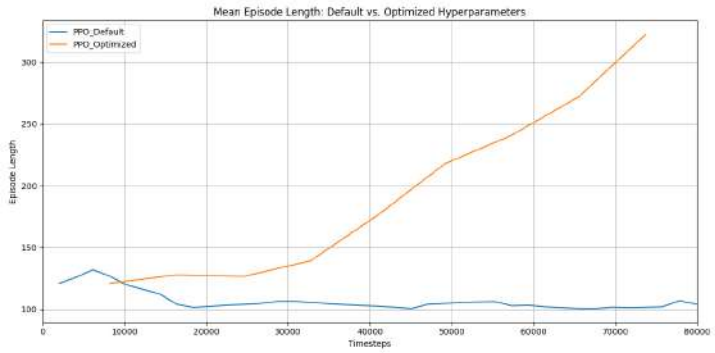

Figure 8. Health Gathering Scenario Mean Episode Length Comparison

increases the rewards received by the agent from the environment.

5.4 Deadly Corridor Scenario Results

This subsection presents the results for the Deadly Corridor scenario achieved by training the PPO agent on the default parameters, provided in Table 1, and the optimized parameters, provided in Table 10. The objective of the Deadly Corridor scenario is to survive from the monsters shooting at the agent in the corridor and reach the vest at the end of the corridor. Both agents are trained at Doom skill level 1 to allow the agents to easily explore the environment without immediately being eliminated by the monsters present in the corridor. The results of training the agent on the default parameters and on the optimized parameters for 100,000 timesteps are shown in Figure 9 and Figure 10.

Figure 9 and Figure 10 indicate that the agent trained on the default parameters performs significantly better than the agent trained on the optimized parameters.

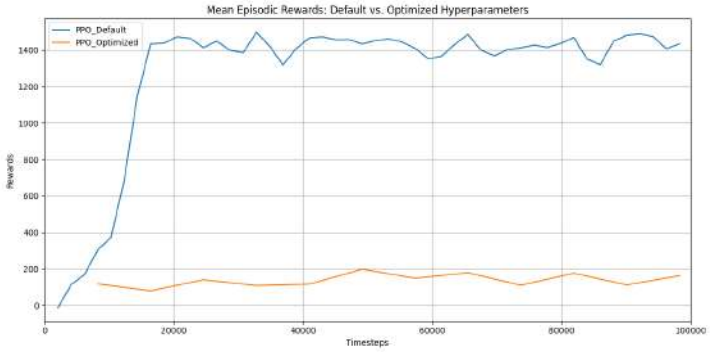

Figure 9. Deadly Corridor Scenario Mean Episodic Rewards Comparison

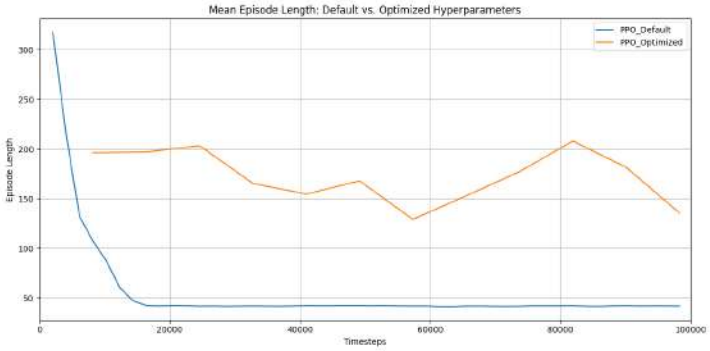

Figure 10. Deadly Corridor Scenario Mean Episode Length Comparison

Both the PPO agents are evaluated for 10 episodes in the Deadly Corridor Scenario on Doom skill level 1. On training the PPO agent by using the default parameters, the agent's performance achieves a mean reward of 1232.2 with a std reward of 532.2; on the other hand, using the tuned parameters for training the PPO agent, its performance gets a mean reward of 28.5 with a std reward of 176.4.

However, when observing the behaviour of the default agent, it is indicated that the higher rewards obtained by this agent were due to an undesirable behaviour, where it learned to exploit the rewards obtained from the environment by ignoring all the enemies present in the corridor and directly rushing towards the vest at the end of the corridor. This behaviour of the PPO agent may provide it with high rewards in environments where the Doom skill level is lower; however, when the skill level is increased, it fails to reach the vest at the end of the corridor and is killed by the monsters as they deal increased damage to the agent.

In comparison, the agent trained on the optimized parameters learned to kill the monsters present in the environment first, indicating that the agent inherits the desired behaviour. Though the reason for getting lower rewards by the PPO agent is due to the time it takes to kill all the monsters, which results in being killed by them. Since the environment of the deadly corridor is complex and requires the agent to handle multiple tasks at once, including navigation towards the vest, combat with the enemies that constantly shoot at the agent, and managing resources such as health and ammunition, the agent is further trained for a total of 550,000 timesteps using curriculum learning, detailed in Section 3.3.5.

The training of the agent is performed for 170,000 timesteps at skill level 1, which includes 100,000 timesteps used for initial training. At skill level 2, the agent undergoes training for 40,000 timesteps. At skill level 3, the agent's training runs for 40,000 timesteps, and the same number of timesteps at skill level 4. The training process runs for 260,000 at the highest difficulty level 5. The default skill level is 5 for this scenario, and is also recommended. The results for training the agent for 550,000 timesteps on different skill levels are presented in Figure 11 and Figure 12.

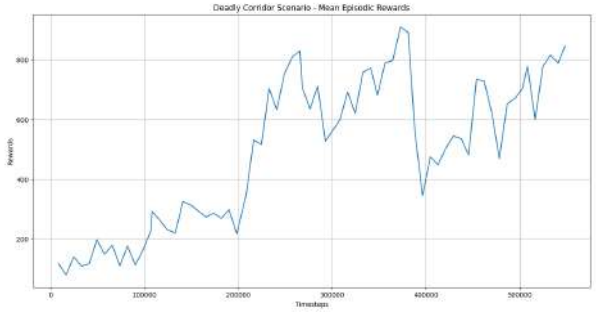

Figure 11. Deadly Corridor Scenario Mean Episodic Rewards for Optimized Hyperparameters

Using the optimized parameters, the PPO agent successfully learns an efficient policy where it eliminates all the monsters and then reaches the vest, which results in an increase in the mean episodic rewards as the agent's training progresses at higher skill levels. Initially, the mean episode length is high, but as the training progresses, the mean episode length decreases and the

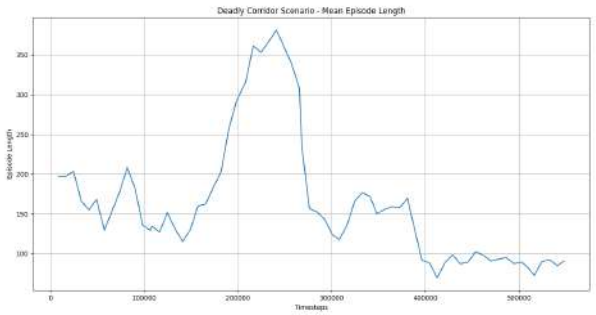

Figure 12. Deadly Corridor Scenario Mean Episode Length for Optimized Hyperparameters

agent learns to eliminate the enemies efficiently and navigate to the vest faster. By the end of the training, the agent shows stable performance across each skill level and achieves higher rewards in the environment, indicating that it has generalized well to the different skill levels of the scenario. After the completion of the training of the PPO agent on optimized parameters for 550,000 timesteps, the agent is evaluated for 10 episodes in the Deadly Corridor scenario on Doom skill level 5. It achieved a mean reward of 840.8 and a std reward of 762.9, while the default agent, evaluated for 10 episodes in the Deadly Corridor scenario on skill level 5, achieved a mean reward of 276.83 with a std reward of 233.01. However, the default agent could not reach the vest in the skill level 5 environment.

These results show that hyperparameter optimization allows the PPO agent to learn an optimal and stable policy, and applying curriculum learning can make the agent learn a policy that can also generalize to environments with higher difficulty levels.

6 Discussion

The Hyperparameter optimization was conducted using Optuna for all four scenarios, as detailed in Section 3.6. The set of hyperparameters chosen for optimization was different for each scenario. The resultant parameters obtained by hyperparameter tuning for each scenario are summarized in Table 11.

Table 11 shows that hyperparameter tuning has a significant impact on the training of the PPO agent. In simple scenarios, such as Basic and Defend The Center, the improvements are small in scale, resulting in a 3.9% increase and a 12.4% increase in the performance of the optimized PPO agent in the Basic and Defend The Center scenarios, respectively. This indicates that the default parameters provided by SB3 are already suitable for these scenarios, where the tasks are very straightforward. This is similar to the findings of [28], where the agent maintained a stable performance in these environments. By comparing the impact of hyperparameter tuning between scenarios, it is highly noticeable in more complex scenarios like Health Gathering and Deadly Corridor, which require multiple skills by the RL agent, such as navigation, combat, and resource management, to complete the objectives. The hyperparameter optimization led to a massive 523.7% increase and a 203.7% increase in the Health Gathering and Deadly Corridor scenario, respectively. The tuned parameters provide the optimized PPO agent with a policy that is much better and more stable than the one learned by the default agent. When the skill level is set to 1 in the Deadly Corridor scenario, the performance of the optimized agent gets worse than the default one; as a result, its performance decreases to 97.6%. Further evaluation of the default agent showed that the decline is due to the default agent exploiting the environment by rushing towards the vest while ignoring all the enemies, which increases the rewards it receives from the environment, whereas the optimized agent kills the enemies first. Although this strategy works for skill level 1, the default

Table 11. Comparison of Rewards Before and After Hyperparameter Tuning

Scenario	Rewards Before Hyperparameter Tuning	Reward After Hyperparameter Tuning	Improvement
Basic	79.1	82.2	3.9%
Defend The Center	12.1	13.6	12.4%
Health Gathering	293.6	1831.4	523.7%
Deadly Corridor (Skill Level 1)	1232.2	28.5	-97.6%
Deadly Corridor (Skill Level 5)	276.8	840.8	203.7%

agent fails to reach the vest in scenarios where the skill level is higher, resulting in a mean reward of just 276.8 in skill level 5. Whereas the optimized agent learns a more generalized policy, which allows it to perform efficiently at higher skill levels, receiving a mean reward of 840.8 in skill level 5. This results in a 203.7% increase in the performance of the optimized agent. All this displays that the hyperparameter tuning provides the RL agent a way to learn a policy that makes it generalize very well in different skill levels of the scenario.

These results indicate that hyperparameter tuning has a limited impact in environments that are simple and involve the agent carrying out straightforward tasks, such as the Basic and Defend The Center scenario, whereas it has a significant impact in environments that involve the agents accomplishing multiple tasks, which increases the complexity of the environments, such as the Health Gathering and the Deadly Corridor scenario. The right hyperparameter can enhance the agent's performance and its stability to learn the optimal policy.

As ViZDoom is a popular and widely adopted RL research platform, we analyzed our findings with earlier reported benchmark results based on DQN and A3C. Mean episodic rewards of 1262 have been reported in experiments with DQN for the health gathering scenario, with more advanced variants reaching rewards of 1890. Using A3C, the mean episodic rewards have been reported as 1578 with advanced variants reaching 2078. In the current research, the optimized PPO agent exhibited a mean episodic reward of 1831.4, which demonstrates competitive performance relative to the previously reported approaches. For the Deadly Corridor scenario, previous research based on reward shaping and curriculum learning report rewards around 1600 at higher difficulty levels. The results of the optimized PPO agent in the current research are comparable with these findings [10, 11, 13, 14, 17] .

Although research continues to use ViZDoom for benchmarking RL agents, with researchers also exploring alternate training strategies such as imitation learning and improved exploration mechanisms for agents [29–33], it is important to note that reward structures and training configurations vary across each study. The currrent study employs customized rewards shaping and curriculum learning strategies to facilitate learning in complex environments. Our configurations affect the reward scale and limit an absolute comparison. The discussed comparison provides a useful contextual understanding demonstrating that the optimzed PPO agent achieves a performance comparable to the existing approaches while maintaining the main objective of the current research which was to analyze the impact of hyperparameter optimization on PPO agents across environments with increasing complexity.

7 Challenges and Limitations

Training the RL agent in the Deadly Corridor scenario was the biggest challenge faced throughout this research. The challenge was the policy learned by the RL agent. Through that policy, the RL agent ignores all the enemies who are shooting at the RL agent, and the agent tries to rush directly towards the vest, which is placed at the opposite end of the corridor. Curriculum learning helps to overcome this challenge, which trains the agent first on the lowest difficulty level, then gradually increases it to its maximum level. Another technique used to address this challenge was reward shaping, where the reward structure of the environment was changed to include rewards for killing the enemies in the corridor, encouraging the agent to eliminate all enemies before reaching the vest. Finally, hyperparameter tuning was performed to allow the agent to learn a policy that efficiently balances navigation, combat, and resource management, increasing the rewards received by the PPO agent from the environment.

This research was limited to training the agent only on the PPO algorithm. A comparison of the PPO algorithm with other popular RL algorithms may provide a better understanding of the agent's performance trained on multiple different algorithms. We employed the default NatureCNN architecture provided by SB3 during the training of all the agents in different scenarios. This limits the findings to a specific CNN architecture for the RL agent, and also keeps away from experimenting with alternative CNN architectures, which may decrease the required computational resources for training the RL agent. Despite these challenges and limitations, this research successfully demonstrates the effectiveness of using the PPO algorithm to train RL agents in complex 3D environments.

8 Conclusion

The study successfully demonstrates the effectiveness of using the PPO algorithm to train RL agents in complex 3D environments and highlights the impact of hyperparameter tuning on the performance of the agents. The results show that the PPO agent was successfully able to learn optimal policies in simple environments, such as Basic and Defend The Center scenarios, using Default parameters, leading to a mean episodic reward of 79.1 and 12.1, respectively. But the agents struggle to achieve optimal behaviour in more complex environments using the default parameters, achieving a mean episodic reward of 293.6 in Health Gathering and 276.8 in Deadly Corridor (Skill Level 5). Performing Hyperparameter optimization greatly increased the performance of the agents in different scenarios. In simple scenarios, such as Basic and Defend The Center, the improvement was moderate, resulting in mean episodic rewards of 82.2 in Basic and 13.6 in Defend The Center, whereas in complex environments, hyperparameter optimization led to a mean episodic reward of 1831.4 in Health Gathering and 840.8 in the Deadly Corridor scenario.

These findings indicate that the systematic hyperparameter optimization of the PPO algorithm using Optuna leads to significant improvements in the performance of the agents, particularly in complex environments that involve multiple tasks like navigation, combat, and resource management, like the Deadly Corridor scenario, resulting in an increase of 203.7% in the performance of the agent. Whereas in simple scenarios, like Basic and Defend the Center, it showed modest improvements of 3.9% and 12.4% respectively. The most notable improvement was in the Health Gathering scenario, where the agent was able to achieve an increase of 523.7% in its performance. The research successfully demonstrates that RL agents trained on the PPO algorithm can learn optimal policies in complex 3D environments involving multiple challenging tasks when trained using appropriate RL techniques and a proper set of hyperparameters.

The study shows that the DRL agents trained on the PPO algorithm, combined with reward shaping and curriculum learning, can learn to behave optimally in complex environments, performing quick planning and strategic decision making. These findings are of critical importance and highly applicable to real-world application domains.

Author Contributions

Areej Fatemah Meghji: Conceptualization, Methodology, Writing - Original draft preparation, Writing - Editing and review, Validation, Formal Analysis, Result reporting. **Rashid Hussain**: Writing - Editing and review, Visualization, Formal Analysis. **Mirza Muhammad Abbas**: Conceptualization, Methodology, Software, Writing - Original draft preparation, Result reporting. **Abdul Lahad**: Methodology, Writing - Original draft preparation, Software, Result reporting, Visualization.

Compliance with Ethical Standards

The authors declare that they have no conflict of interest. It is also declared that this article does not contain any studies with human participants or animals. The data supporting the findings of this study has been generated from experiments conducted using the ViZDoom reinforcement learning environment. The implementation used in this study is based on the reinforcement learning library Stable-Baseline3 with hyperparameter optimization performed using Optuna.

Funding Information

No external funding was received for this study.

References

[1] Z. Li, Q. Ji, X. Ling and Q. Liu, "A comprehensive review of multi-agent reinforcement learning in video games," *IEEE Transactions on Games*, 2025.

[2] M. A. Mirza, A. Lahad, and A. F. Meghji, "A Study of Q-Learning in the Taxi-v3 Environment: Reinforcement Learning for Optimal Navigation through Hyperparameter Optimization," *KIET Journal of Computing & Information Sciences"*, vol. 8, no. 1, pp. 54-65, 2025. https://doi.org/10.51153/w4rsbd31

[3] R. S. Sutton, and A. G. Barto, "Reinforcement learning: An introduction," vol. 1, no. 1, pp. 9-11. Cambridge: MIT press, 1998.

[4] C. J. Watkins, and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp.279-292, 1992.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.

[6] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation", *Advances in neural information processing systems*, 12, 1999.

[7] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, and R. Józefowicz, "Dota 2 with large scale deep reinforcement learning," arXiv preprint arXiv:1912.06680, 2019.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," In International conference on machine learning (pp. 1928-1937). PmLR, 2016.

[9] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization (TRPO)," CoRR abs/1502.05477, 2015.

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

[11] A. Khan, and M. Naeem, M. "Evaluating reinforcement learning algorithms in first-person shooter games using VizDoom," *Multimedia Tools and Applications*, vol. 84, no. 15, pp. 15053-15075, 2025.

[12] M. Wydmuch, M. Kempka, and W. Jaśkowski, "Vizdoom competitions: Playing doom from pixels," *IEEE Transactions on Games*, vol. 11, no. 3, pp.248-259, 2018.

[13] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining pp. 2623-2631, 2019.

[14] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," In 2016 IEEE conference on computational intelligence and games (CIG) (pp. 1-8). IEEE, 2016.

[15] J. Karttunen, A. Kanervisto, V. Kyrki, and V. Hautamäki, "From video game to real robot: The transfer between action spaces," In ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 3567-3571). IEEInternational Symposium.

[16] A. Zzakhakharenkov and I. Makarov, "Deep reinforcement learning with dqn vs. ppo in vizdoom," in 2021 IEEE 21st international symposium on computational intelligence and informatics (CINTI), pp. 000131–000136, IEEE, 2021.

[17] G. Lample, and D. S. Chaplot, "Playing FPS games with deep reinforcement learning," In Proceedings of the AAAI conference on artificial intelligence, 2017.

[18] M. Kalra and J. Patni, "Playing doom with deep reinforcement learning," *Recent Trends in Science, Technology, Management and Social Development*, pp. 42, 2018.

[19] W. Ali, S. Lakho, N. N. Bhatti, and I. A. Memon, "Adaptive Bug Localization Framework for Precision-Driven Bug Localization in Software Engineering," *VFAST Transactions on Software Engineering*, vol. 12, no. 3, pp. 230–242, 2024.

[20] M. R. Hossain and D. Timmer, "Machine learning model optimization with hyper parameter tuning approach," *Glob. J. Comput. Sci. Technol. D Neural Artif. Intell*, vol. 21, no. 2, p. 31, 2021.

[21] J. a Ilemobayo, O. Durodola, O. Alade, O. J. Awotunde, A. T. Olanrewaju, O. Falana, A. Ogungbire, A. Osinuga, D. Ogunbiyi, A. Ifeanyi, et al., "Hyperparameter tuning in machine learning: a comprehensive review," *Journal of Engineering Research and Reports*, vol. 26, no. 6, pp. 388–395, 2024.

[22] M. Towers, A. Kwiatkowski, J. Terry, J. U Balis, G. De Cola, T. Deleu, M. Goul~ao, A. Kallinteris, M. Krimmel, A. KG, et al. "Gymnasium: A standard interface for reinforcement learning environments," arXiv preprint arXiv:2407.17032, 2024.

[23] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *Journal of Machine Learning Research*, vol. 21, no. 181, pp. 1-50, 2020.

[24] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, and S. Ghemawat, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," arXiv preprint arXiv:1603.04467, 2016.

[26] Google, "Google Colaboratory", 2026. Retrieved January 03, 2026, from https://colab.research.google.com/

[27] M. A. Mannan, R. Qamar, I. U. Khan, A. Hussain, S. Ahmed, and J. Khan, "Evaluating the Performance of Machine Learning Classifier Algorithms for Software Estimation in Software Development Projects," *VFAST Transactions on Software Engineering*, vol. 12, no. 1, pp. 70–78, 2024.

[28] H. Bhuwad, R. Nikam, and D. L. S. Gunjal, "Optimizing DOOM Game Using Reinforcement," *In Proceedings of the International Conference on AI and Robotics: AIR 2025*, vol. 1, pp. 374. Springer Nature, 2025.

[29] A. Khan, A. A. Shah, L. Khan, M. R. Faheem, M. Naeem, and H. T. Chang, "Using vizdoom research platform scenarios for benchmarking reinforcement learning algorithms in first-person shooter games," *IEEE Access*, 12, 15105-15132, 2024.

[30] A. Khan, and A. Aqeel, "Benchmarking reinforcement learning algorithms in first-person shooter games using VizDoom," *Entertainment Computing*, 101031, 2025.

[31] R. Spick, T. Bradley, A. Raina, P. V. Amadori, and G. Moss, "Behavioural cloning in vizdoom," arXiv preprint arXiv:2401.03993, 2024.

[32] C. Zhang, H. Hu, Y. Zhou, X. Wang, and E. S. Liu, "HIFAS: A Hybrid Interactive FPS Agent System for Large Game Maps," *IEEE Transactions on Games*, 2025.

[33] S. S. Wagner, and S. Harmeling, "Just cluster it: An approach for exploration in high-dimensions using clustering and pre-trained representations," arXiv preprint arXiv:2402.03138, 2024.