# Curriculum Learning and Policy Proximal Optimization in the Pong Game

Alekha Rao and Ankitha Raman

**Abstract**

Pong is a classic and popular arcade game in which players control vertical paddles to deflect a ball back and forth across the screen. The objective is to defeat the other player so that they are unable to hit the ball back. Inspired by the Atari game environment with its strategic depth and simplicity due to its discrete action space, we developed a custom Pong-like environment designed to facilitate curriculum-based reinforcement learning. In this work, we propose a training framework that combines Curriculum Learning with Proximal Policy Optimization (PPO) to teach an agent to play Pong efficiently and effectively. Instead of learning the full game dynamics all at once, the agent progresses through a series of five increasingly difficult stages—beginning with basic ball-tracking and ending in high-speed ball movements with precise agent gameplay. PPO's clipped objective ensures stable policy updates across stages. We evaluate agent performance using cumulative, per-episode, and normalized reward metrics, and find that our curriculum approach significantly improves learning rate compared to non-curriculum PPO baselines. These results demonstrate the advantages of structured task decomposition and a customized environment design in training agents for complex behaviors in sparse-reward settings.

## 1 Introduction

Training an agent to play an online video game has been a significant focus within the fields of artificial intelligence and reinforcement learning (RL). Classic games like Pong are often used as benchmarks because of their simplicity, discrete action space, and focus on timing and control. In this project, we design our own Pong-inspired environment using the custom gym template [2] rather than using the standard Atari version. Our environment removes the second paddle (opponent), giving the agent a more controlled learning experience, and includes custom reward functions to make feedback less sparse and more informative. This setup gives us more flexibility in shaping the agent's learning process and allows for easier debugging and performance evaluation.

Even though Pong is simple in concept, training an agent to play it well can still be difficult due to delayed rewards and the need for precise movement.

To address this, we use a combination of Curriculum Learning and Proximal Policy Optimization (PPO). By breaking down the full task into smaller subtasks, updating the policy more frequently, and increasing difficulty gradually, we give the agent a smoother learning curve. PPO helps by making stable and frequent updates to the policy throughout training. Together, these methods help improve learning efficiency and overall agent performance in our custom Pong environment.

# 2 Background Related Work

## 2.1 Technical Background

Curriculum Learning is a training strategy in reinforcement learning where an agent is progressively introduced to more difficult tasks instead of learning the whole task at once. Inspired by the way humans learn by mastering simpler tasks before tackling complex challenges, Curriculum Learning helps improve sample efficiency and stability in RL training. The design of a curriculum involves two main components: a Difficulty Measurer and a Training Scheduler. The Difficulty Measurer defines task complexity based on predefined rules such as task complexity or environmental parameters. The Training Scheduler determines the sequence and pacing at which the model progresses through tasks, which can predefined manually or adaptively adjusted based on performance [10]. A crucial aspect of Curriculum Learning is designing source tasks, which serve as stepping stones toward mastering the final objective. Some examples of possible methods for defining source tasks are mistake-driven subtasks, task-based subgoals, and task dimension simplification [7]. In particular, Continuous Curriculum Learning extends this idea by adjusting the difficulty of tasks in a smooth and ongoing fashion rather than in discrete stages, allowing the agent to transition more fluidly between skill levels [1]. By carefully designing source tasks, Curriculum Learning ensures that reinforcement learning agents acquire skills in a structured manner, leading to improved generalization and better performance in complex tasks.

Proximal Policy Optimization (PPO) is a reinforcement learning algorithm that belongs to the family of policy gradient methods [9]. Unlike value-based methods such as Deep Q-Networks, PPO directly optimizes the policy by adjusting the probability distribution of actions taken by the agent. This approach is particularly effective for complex environments where traditional value-based methods struggle. One of PPO's key innovations is its clipped surrogate objective function, stabilizing training by preventing excessively large policy updates [9]. In standard policy gradient methods, large updates can cause instability and suboptimal performance. PPO mitigates this issue by limiting how much the policy can change each update, which ensures controlled learning. When combined with Curriculum Learning, PPO can progressively refine the agent's strategy, leading to more efficient training and improved performance. PPO has

been successfully applied to a wide range of tasks, including Atari game environments, where it consistently demonstrates strong performance[4]. Its ability to handle high-dimensional observation spaces and continuous action spaces makes it a popular choice in many modern reinforcement learning benchmarks.

## 2.2 Related Work

Previous research has explored training agents to play Atari Pong; however, these approaches have primarily focused on Deep Reinforcement Learning [5] and Hierarchical Reinforcement Learning [3] rather than Curriculum Learning or Proximal Policy Optimization. While Deep RL has been widely used to train agents for Atari games, Hierarchical RL has introduced structured learning through sub-task decomposition. Understanding these existing approaches provides a foundation for exploring alternative reinforcement learning strategies.

**Deep Reinforcement Learning:**
Deep RL has gained traction as a methodology for solving complex reinforcement learning problems, particularly with advancements in neural networks and computer vision. Prior applications of Deep RL to Atari games have involved the use of deep convolutional neural networks combined with updates from the Deep Q-learning algorithm [5]. This algorithm leverages stochastic gradient descent to approximate the action-value function, allowing Deep Q-Networks (DQNs) to process high-dimensional visual input and learn effective feature representations. These DQNs were tested on a wide variety of Atari games, including Pong, and they were able to achieve and sometimes even exceed the human-level performance in many of the games [6]. Overall, this approach has proven successful in optimizing policies for agents playing various Atari games.

**Hierarchical Reinforcement Learning:**
The core idea of Hierarchical RL is that a complex task can be decomposed into smaller, more manageable sub-tasks, which are then learned and combined into an overall policy. By incorporating human-like instructional materials—such as game manuals or direct human guidance— Hierarchical RL has been shown to accelerate the agent's learning process [3]. This method enables agents to develop sub-policies for each sub-task, ultimately leading to improved agent performance. Interestingly, integrating human-like instructions into Hierarchical RL has been integral in enhancing learning efficiency for Atari games like Tennis and Pong, outperforming Deep RL in these contexts [3].

While Hierarchical RL shares similarities with Curriculum Learning—both break down a challenging task into smaller components— their fundamental approaches differ. Hierarchical RL employs multiple levels of abstraction where task structures and state representations may vary across different hierarchical layers. In contrast, Curriculum Learning is specifically applied during training and focuses on optimizing the sequence in which tasks are presented [10]. Understanding these distinctions is crucial for evaluating the potential benefits of Curriculum

Learning in reinforcement learning for our custom Pong environment.

# 3 Technical Approach

## 3.1 Proximal Policy Optimization

The key principle behind PPO is to prevent excessively large updates that could destabilize the policy. To achieve this, PPO optimizes the following objective function:

$$L(s,a,\theta_k,\theta) = \min(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s|a), clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon) A^{\pi_{\theta_k}}(s|a)),$$

where $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$ is the probability ratio between the new and old policy. $A^{\pi_{\theta_k}}$ is the advantage estimate, which measures how much better an action is compared to the expected value. And $\epsilon$ is a clipping hyperparameter that controls how far the new policy can deviate from the old one [8]. The clipping mechanism ensures that if the policy update leads to a probability ratio that is too large, it gets clipped to avoid instability. This prevents the policy from moving too far away from the previous versions while still making necessary improvements.
The policy updates as follows:

$$\theta_{k+1} = \arg\max_\theta \mathbb{E}_{s,a\sim\pi_{\theta_k}} \left[ L(s,a,\theta_k,\theta) \right],$$

where the optimization is performed by taking multiple steps of minibatch SGD to maximize the objective [8].
The algorithm for PPO is as follows:

**Algorithm 1** PPO-Clip

1: Input: initial policy parameters $\theta_0$, initial value function parameters $\phi_0$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function $V_{\phi_k}$.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \min\left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t,a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t,a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k|T} \sum_{\tau\in\mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

Figure 1: Psuedocode for PPO algorithm using stochastic gradient ascent [8]

### 3.2 Curriculum Learning Breakdown

To make learning more efficient, we gradually increase the difficulty of the environment in five stages. ***Note:*** The dimensions of the screen range from 0-1, with the origin (0,0) being the bottom left corner of the screen.

**Stage 1**
Goal: Can the agent track and follow the ball?

- Ball moves only in the vertical direction
- Slow ball speed ($v_y = 0.2$)
- Large paddle size (0.5)

**Stage 2**
Goal: Can the agent hit the ball back?

- Ball moves in both x and y directions
- Slow ball speed ($v_x = v_y = 0.2$)
- Large paddle size (0.5)

**Stage 3**
Goal: Can the agent react faster to the ball?

- Experiment with faster ball speeds (when the ball is reset, $v_x$ and $v_y$ will be randomly set to either -0.2, -0.1, 0.1, or 0.2).

**Stage 4**
Goal: Can the agent perform well when the ball is in different positions?

- Vary the ball's initial position (when the ball is reset, $ball_x$ and $ball_y$ will be randomly set anywhere in the range from 0.1-0.9 in 0.1 increments).

**Stage 5**
Goal: Can the agent make more precise movements?

- Experiment with smaller paddle sizes (every 100 episodes, the paddle size will decrease by 0.05 starting at an initial size of 0.5 and ending at 0.3).

## 4 Experimental Results

To evaluate the performance of our PPO agent trained with Curriculum Learning in the Pong environment, we visualized key metrics across three different plots for each stage of training. These plots were designed to capture different aspects of the agent's learning and progression over time. First, we plotted the cumulative reward vs. total timesteps, which provides an overall measure of the agent's learning progress by summing rewards across the entire training

process. Second, we plotted the total reward per episode, which highlights fluctuations in performance at the episode level. Lastly, we plotted the normalized reward per episode, which reflects the average reward per timestep within each episode—offering insight into the agent's efficiency while also accounting for varying episode lengths. In addition to training with Curriculum Learning, we also trained the agent using PPO alone without any staged curriculum, serving as a baseline for comparison. By analyzing the same three plots for both the curriculum-based and baseline agents, we were able to directly assess the impact of Curriculum Learning on the agent's ability to learn more efficiently, generalize more effectively, and maintain high performance across increasingly complex stages.

## 4.1 Curriculum Learning Performance by Stage

**Stage 1: Learning Basic Ball Tracking**
The objective of Stage 1 was to assess whether the agent could learn to track and follow the ball under simplified dynamics. The training curves shown in

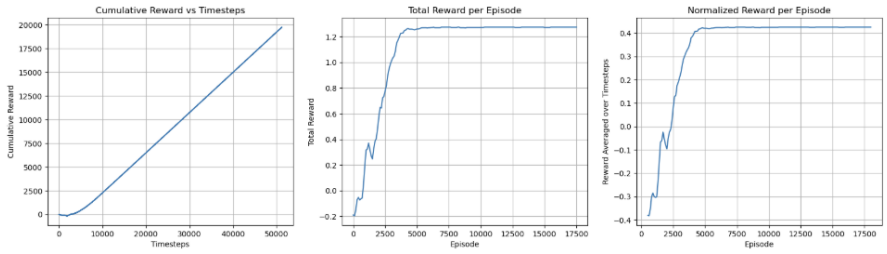


Figure 2: Stage 1 Training Performance

Figure 2 reflect successful learning in Stage 1. In the cumulative reward vs. timestep plot, we observe that initially the reward starts slightly negative but then steadily increases. This suggests that the agent initially struggled, but then quickly acquired the ability to follow the ball as it moved. Additionally, the second plot shows similar growth over time, followed by stabilization at a reward of approximately 1.3 per episode. This is also evident in the normalized reward plot, where the agent's performance improved over time until it eventually stabilized at around 0.4 per timestep. These plots demonstrate rapid convergence, indicating that the agent was able to master this simple task relatively quickly. Overall, the results from Stage 1 showed that the simplified environment was effective in teaching the agent basic ball-tracking behavior, setting the foundation for the later, more complex stages.

**Stage 2: Learning to Hit the Ball Back**
In Stage 2, we increased the environmental complexity to evaluate whether the agent could learn to return the ball successfully when it moves in two dimensions. These training curves in Figure 3 reveal a clear struggle, followed by

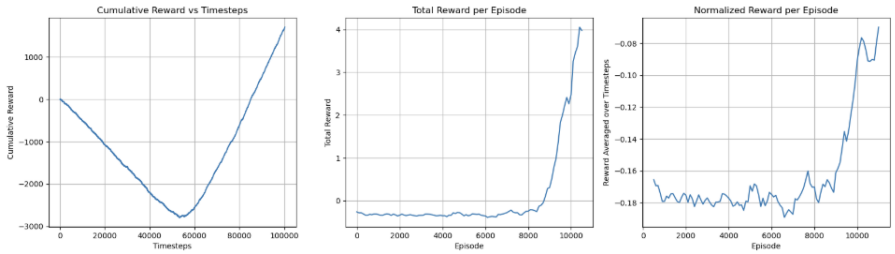


Figure 3: Stage 2 Training Performance

a breakthrough in learning. In the cumulative reward over timesteps plot, we observe a sharp decline into negative cumulative reward, indicating that the agent consistently missed the ball for a significant portion of the training. This continued until around 70,000 timesteps, at which the agent began to recover and accumulate positive reward, displaying a turning point in its learning development. The total reward per episode plot mirrors this learning curve as we can see that for most of the training, the agent's episode rewards hover near zero, but in the final stretch of training episodes, we observe a rapid improvement. This is a strong sign that the agent has finally learned to track the ball and return it effectively. Similarly, in the normalized reward per episode plot, the agent's efficiency remains poor for most of the run but improves at the end, indicating growing reward per timestep. Additionally, in comparison to Stage 1, during Stage 2, the agent went through far fewer episodes, meaning that episode lengths began to increase. This also shows learning because the agent wasn't dying as often and was learning how to better play the game. This stage demonstrates that introducing horizontal ball movement posed a major challenge; however, the agent eventually developed the skill of hitting the ball back.

### Stage 3: Reacting to Faster Ball Speeds

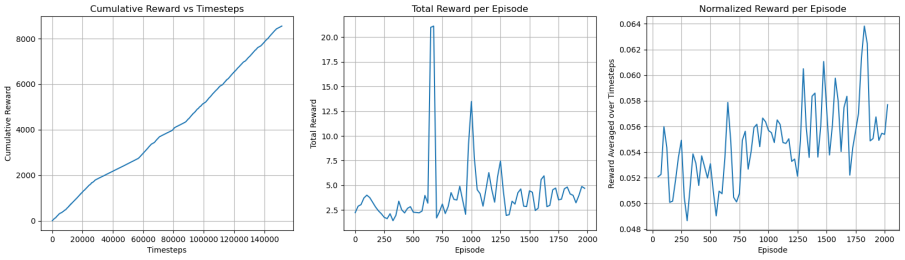


Figure 4: Stage 3 Training Performance

In Stage 3, we experimented with the ball speed, forcing the agent to react more quickly. The plots in Figure 4 indicate a general improvement in learning, although the trends present are more subtle than those in the first two stages. The cumulative reward over timesteps plot demonstrates an almost linear increase, suggesting that the agent was relatively consistent in its performance of hitting the ball back at higher ball speeds. In terms of total reward per episode, the agent started earning a reward of about 2.5 and ended up receiving a reward close to 5 (doubling its reward) by the final episode. While not as stable as the last stage, the agent's maximum reward increased. In addition, there were periods of a few episodes where the agent outperformed itself (as shown by the peaks in the graph at episodes 700, 1000, and 1250) and earned a much higher reward (at about 21, 14, and 7.5, respectively) than in the average episode. It's also important to notice that 2000 episodes ran in this stage in comparison to over 10,000 in Stage 2, emphasizing that the agent is playing longer games and interacting with the ball more in each successive stage. Finally, while the normalized reward over timesteps per episode plot shows a decent bit of noise, we can observe a general upward trend, indicating that the agent is learning throughout this stage and is getting more reward at each timestep as the episodes proceed and the ball speed varies.

**Stage 4: Generalizing to Random Ball Positions**

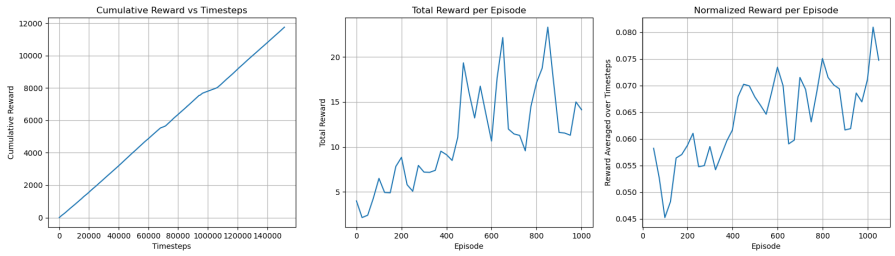


Figure 5: Stage 4 Training Performance

In Stage 4, we tested the agent's ability to generalize to varied initial ball positions. This stage was crucial for evaluating the agent's flexibility and ability to adapt to new situations. The training curves in Figure 5 indicate that the agent adapted well to this new challenge. The cumulative reward over timesteps plot shows a near-linear upward trajectory, indicating stable learning despite the newly added challenge. This indicates that the agent was able to effectively transfer its learning from previous stages to adapt to this new environment. In the total reward per episode plot, we observe a generally increasing trend with some variance across episodes. This variance is expected due to the randomized nature of the environment, which introduces inconsistency in difficulty from episode to episode. However, the overall rise in episode rewards indicates

that the agent's policy is effective across various initial conditions. The normalized reward per episode plot also displays a gradual upward trend, suggesting that the agent is becoming more efficient in accumulating reward relative to the number of timesteps. We also note that the number of episodes in this stage has been reduced to 1000, which is half the number we had in Stage 3. This emphasizes that the agent is surviving longer in each episode, suggesting improved gameplay as it successfully adapted to the challenge of randomized ball positions. Overall, Stage 4 demonstrates that the agent is learning to generalize its behavior and maintain consistent performance regardless of where the ball starts.

**Stage 5: Making Precise Movements**


Figure 6: Stage 5 Training Performance

Stage 5 was the final stage in our Curriculum Learning plan, and was used to test the agent's ability to make precise movements. The cumulative reward vs timesteps plot shows a steady, nearly linear increase throughout the stage, suggesting the agent was able to consistently accumulate reward even as paddle size decreased. This suggests that the agent's policy was robust enough to withstand incremental reductions in the paddle size. The total reward per episode plot reveals more nuanced behavior. While there is significant fluctuation in episode rewards, the agent is generally able to recover and maintain performance. While performance fluctuates, there is no collapse in reward, indicating that the agent maintains a stable level of ability even as the environment becomes more precise. Despite the variance, the normalized reward per episode plot demonstrates relative stability and even slight improvement over time. This indicates that the agent maintained efficiency in reward collection per timestep, despite the increasing precision needed with smaller paddle sizes. Additionally, the total number of episodes decreased slightly from the previous stage. This indicates that even with the increased precision demands, the agent was able to maintain long gameplay, reflecting improved control.

## 4.2 Baseline PPO Performance and Comparison

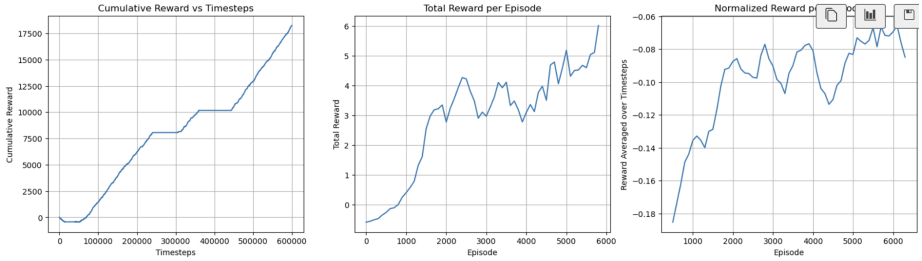


Figure 7: Agent Training Performance under PPO

Figure 7 shows an agent operating solely under the PPO algorithm but without Curriculum Learning during its training process. Essentially, this agent is performing in the same environment that the CL agent performed in during Stage 5. This agent performed over 600,000 timesteps, matching the CL agent's total timesteps over five stages.

In terms of cumulative reward vs. timesteps, the agent initially received negative reward for the first 70,000 timesteps or so, meaning that the agent found it difficult to adjust to the Pong game environment and hit the ball back. Soon after, however, the agent picked up the game and demonstrated a relatively steady increase in cumulative reward (with the exception of two plateaus at about 235,000 and 350,000 timesteps). At the end of the 600,000 timesteps, the agent accumulated over 17,500 in reward, while the CL agent at the end of Stage 5 accumulated over 12,000 in reward after 150,000 timesteps.

For total reward per episode, the agent demonstrated growth over its 6000 episodes, with a maximum of 6 by the last episode. Although this total reward is not particularly high, especially in comparison to the average of between 15-20 during Stage 5, we can observe the agent learning a lot during training. Lastly, we have the plot showing the normalized reward over episodes, which accounts for episodes that have varying lengths. On average, the agent initially received a reward of -0.18, which increased to about -0.07 on the final episode. Although improvement is clear, the agent maintained a negative normalized reward across all training episodes, indicating that its performance is still suboptimal.

All three plots in Figure 7 have significantly lower results than the plots in Figure 6 (the Curriculum Learning agent during the fifth stage) even though both agents are given the same amount of time to train. These results highlight the key advantage of Curriculum Learning: by progressively increasing the difficulty of the environment and allowing the agent to master simpler subtasks first, Curriculum Learning enables faster adaptation and stronger performance in complex reinforcement learning tasks.

# 5 Conclusion and Future Work

In this project, we successfully demonstrated the effectiveness of combining Curriculum with Proximal Policy Optimization (PPO) to train a reinforcement learning agent in a custom Pong-like environment. By incrementally increasing task difficulty through a five-stage Curriculum Learning, we enabled the agent to first master fundamental skills such as tracking the ball before progressing to more advanced challenges such as making more precise movements. Our staged training approach led to more stable and efficient learning compared to a non-curriculum PPO baseline, as shown by superior performance across cumulative, episodic, and normalized reward metrics. These results highlight how structured task decomposition and environment customization can significantly accelerate learning and improve agent performance, especially in sparse-reward settings.

Despite the promising results, there are several limitations to our current framework. While the curriculum effectively guides the agent toward higher performance, it is manually designed and not dynamically adjusted based on the agent's performance, which could lead to inefficiencies or plateauing in more complex environments. Additionally, our experiments were conducted in a simplified Pong setting with a single agent and no adversarial opponent, which may limit generalization to more competitive or multi-agent scenarios. For future work, we could explore automated Curriculum Learning generation that adapts the difficulty of the stages based on real-time agent performance. Moreover, we could extend this approach to more complex environments with different dynamics and multi-agent interactions. These directions could provide deeper insights into scalable training strategies for real-world reinforcement learning problems.

# References


[1] Andrea Bassich, Francesco Foglino, Matteo Leonetti, and Daniel Kudenko. Curriculum learning with a progression function. *arXiv preprint arXiv:2008.00511*, 2020.

[2] Farama Foundation. Create a custom environment. `https://gymnasium.farama.org/introduction/create_custom_env/`, 2024.

[3] Hua Huang and Adrian Barbu. Playing atari ball games with hierarchical reinforcement learning. *arXiv preprint arXiv:1909.12465*, 2019.

[4] Shogulom Kurganov. Atari games with proximal policy optimization, 2023.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, et al. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, and et al Veness. Human-level control through deep reinforcement learning. *Nature*, 518(7540).

[7] Sanmit Narvekar, Baoxiang Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. 2016.

[8] OpenAI. Proximal policy optimization (ppo) demonstration, n.d.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[10] Xiao Wang et al. A survey on curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.