university of groningen / faculty of science and engineering / bernoulli institute

Safe Deep Reinforcement Learning for Super Mario Bros using Heuristics

Bachelor's Project Thesis

Amir Abdul Aziz, s4019555, a.abdulaziz@student.rug.nl
Supervisors: J.D. Cárdenas Cartagena

This thesis explores the integration of heuristic safety mechanisms within Deep Reinforcement Learning (Deep RL) frameworks to enhance agent survival in the complex, dynamic environment of Super Mario Bros. We examine the influence of heuristic-augmented Actor-Critic models on agent behavior, particularly in terms of safety without compromising learning efficiency. The study compares various configurations, including separate and shared neural network architectures and different combinations of frame stacks for the input layer. It quantitatively measures survival rates, highlighting a significant improvement in avoiding fatal encounters with the most effective configuration around the 10000th episode mark. However, these safety mechanisms may also constrain exploration and long-term reward optimization. Despite achieving a stable survival pattern and a consistent minimum mean loss, a tension between safety and reward maximization is evident. This work contributes to the domain of Safe RL, emphasizing the nuanced trade-offs in stochastic gaming environments.

1 Introduction

Reinforcement Learning (RL) embodies a specific approach within the field of artificial intelligence, aiming to equip agents with the capability to learn optimal behaviors through autonomous interaction with their environment. This learning paradigm, distinguished by its use of a reward function as a feedback mechanism, allows agents to choose actions that maximize cumulative rewards over time. The significance of RL extends beyond theoretical interest, finding practical applications in domains ranging from robotics to complex game environments (Kober, Bagnell, and Peters (2013)).

Video game environments, exemplified by the OpenAI Gym interface introduced by Brockman et al (2016), offer a perfect ground for the application and testing of RL algorithms. These simulated settings provide controlled, yet challenging, arenas where RL agents can be trained, tested, and optimized without the constraints and unpredictability of real-world environments. The inherent continuous and dynamic nature of such games, combined with the essential requirement for real-time decision-making, highlights the critical demand for sophisticated RL strategies tailored to manage complex, sequential decision-making tasks. The actor-critic method, while a significant innovation within the RL framework, was not the first to make impactful strides in video game environments, such as those represented in the OpenAI Gym. It was not until 2016, that adaptations of the actor-critic method, specifically the Asynchronous Advantage Actor-Critic (A2C) algorithm, began to show promise Mnih, Badia, Mirza, Graves, Lillicrap, Harley, Silver, and Kavukcuoglu (2016). These methods were built upon the foundation laid by DQNs, offering an architecture that integrated the strengths of both policy-based and value-based approaches to address the nuanced challenges of such environments. Therefore, while the actor-critic approach now stands as a robust framework for enhancing learning dynamics, it represents a continuing evolution in the search for more efficient and nuanced reinforcement learning processes.

The actor-critic architecture, introduced by Konda and Tsitsiklis (1999) branches the learning process into two interconnected components: the actor, responsible for policy execution by selecting actions, and the critic, which evaluates the actions taken by the actor based on the received reward signal. This division not only accelerates the learning process, by using the value function to adjust the policy directly but also enriches the agent's ability to navigate complex decision spaces, using the critic's evaluation to identify which actions lead to better long-term outcomes. The utility of this method combined with Deep Reinforcement Learning is particularly shown in environments characterized by high-dimensional state and action spaces, such as those encountered in video

games, where traditional RL approaches face scalability challenges. However, optimizing long-term outcomes sets the stage for addressing one of the most critical challenges in Reinforcement Learning: ensuring safety in decision-making.

The integration of Safe Reinforcement Learning (Safe RL) principles, as surveyed by Garcıa and Fernández in their seminal 2015 work, marks a pivotal advancement in the field of Reinforcement Learning, introducing a nuanced layer of safety to the domain. This safety arises from the fundamental requirement that RL agents must not only strive for optimal performance in their learning progress but also strictly adhere to a set of safety constraints. This dual mandate is particularly critical in applications where the agent's actions have real-world consequences, and unforeseen behaviors can result in adverse or even catastrophic outcomes. Applications ranging from autonomous vehicular navigation to robotic surgery and financial trading systems underscore the imperative of integrating safety within the RL framework.

The challenge, as Garcıa and Fernández articulate, lies in maintaining the delicate balance between exploration and exploitation—an intrinsic aspect of Reinforcement Learning—while ensuring that the agent's actions remain within a safe operational envelope. This involves the agent learning to navigate its environment and improve its policy based on trial and error, without engaging in actions that could lead to fatal states.

In this research, we focus on the application of Safe Deep Reinforcement Learning, specifically the actor-critic algorithm, within the intricate video game environment of Super Mario Bros. We will be exploring dual-architecture systems that feature both separate and integrated networks for the actor and critic. We aim to enhance learning efficiency and ensure safe interactions within the game. Moreover, employing heuristics for safety—such as predefined actions when a danger threshold is met—this research navigates the challenges of integrating safety constraints within the high-dimensional, dynamic task, Super Mario.

1.1 Motivation

The interest behind this research stems from the inherent complexity and unpredictability of video game environments, which present significant challenges for the application of Reinforcement Learning. Video games, particularly those with intricate dynamics like Super Mario, require agents to make decisions in real-time within highly variable contexts (Shaker, Togeliu, et al. (2011)). This unpredictability necessitates algorithms that can not only learn and adapt to a wide range of scenarios but also do so safely.

The integration of safety mechanisms within RL algorithms is crucial, as it ensures that the agent's exploration of its environment does not lead to undesirable or fatal outcomes. Such safety measures are especially pertinent in environments where the cost of errors or the unpredictability of situations could significantly impact the learning process or the outcome. This research aims to address these challenges by exploring the application of Safe Deep RL, using the actor-critic method, to navigate the complexities of dynamic game environments while ensuring safe interactions. The necessity for this study is underscored by the potential of Safe RL to enhance the efficacy and applicability of RL in not only gaming but also in broader domains where safety and adaptability are paramount.

1.2 State of the Art in Safe Reinforcement Learning

The current state of the art in Safe RL and actor-critic methods reflects significant advancements and ongoing challenges in applying these techniques to dynamic environments, particularly video games. The actor-critic framework, a cornerstone of modern RL, synergizes policy-based and value-based approaches to enable efficient policy improvement and value function estimation. This dual mechanism is instrumental in complex decision-making games, with Fu, Liu, et al. (2021) providing foundational insights into its operational dynamics. However, the application in more complex unpredictable environments, such as video games, necessitates further innovations to balance learning efficiency with operational safety.

Another useful example of advancements in safe RL, detailed by Berkenkamp, Turchetta, et al. (2017), focuses on ensuring robust performance in the face of environmental uncertainties, introducing models that adaptively manage the exploration-exploitation trade-off while maintaining safety constraints.

The development of Soft Actor-Critic (SAC) algorithms by Haarnoja, Zhou, Hartikainen, et al. (2018) marks a significant leap forward, offering a framework that optimizes policy performance with an emphasis on entropy, thus promoting exploration diversity and improving robustness in varied settings for Actor-Critic.

Despite these advancements, the integration of safety mechanisms within RL applications remains a pressing research gap. The dynamic nature of games, characterized by rapid changes and unpredictable elements, poses unique challenges for RL. Achieving a balance between efficient learning and ensuring safety in such environments is complex. Achiam, Held, Tamar, and Abbeel (2017) have made strides in addressing these challenges by

proposing constrained policy optimization, which aims to incorporate safety constraints directly into the RL optimization process. However, the specific application of these methodologies to video game environments, where the need for real-time decision-making and adaptability to sudden changes is paramount, is still underexplored.

1.3 Contributions of the Thesis

This thesis contributes to the field of Safe Deep Reinforcement Learning (RL) in several key areas. This thesis explores dual-architecture systems for actor-critic methods, comparing separate networks for policy and value functions against a shared, two-headed network architecture. This investigation sheds light on the efficiency and effectiveness of each approach in complex environments like Super Mario.

Moreover, the research integrates heuristic safety mechanisms designed to mitigate risks, enhancing the agent's safety without significantly compromising its learning capability. The heuristic is based on fine-tuning the safe distance to the danger and using that as an indicator of a switch from a policy.

Lastly, through empirical analysis, the study offers novel insights into the suitability of actor-critic methods for navigating a video game, underpinned by a unique, death-focused success metric. This metric is more focused on the factor of safety and therefore would advance our understanding of how to balance learning efficiency with safety considerations in RL applications.

2 Theoretical Framework

2.1 Reinforcement Learning

The foundational concept of Reinforcement Learning is framed within the context of Markov Decision Processes (MDPs), where an agent's objective is to maximize cumulative rewards over time. This cumulative reward is represented by the formula:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{2.1}$$

Where γ denotes the discount factor, balancing the importance of immediate versus future rewards, and r represents the reward received after taking an action at time t and transitioning to the next state. In this setup, the agent iteratively learns the optimal actions to take in given states to maximize its goal. This theoretical framework not only establishes the basis for understanding agent-environment interactions but also sets the stage for exploring more complex learning algorithms. The principle of maximizing future discounted rewards underpins various RL strategies, leading to the development of sophisticated models designed to navigate and learn from complex environments, as actor-critic. This introduction and the gain formula provide a crucial foundation for the subsequent discussion on Temporal Difference Learning, a method pivotal for computing and updating value estimates based on observed rewards and anticipated future rewards.

2.2 Temporal Difference and Reward Computing

Building on the foundation of RL, Temporal Difference (TD) Learning stands out as a crucial mechanism for iteratively updating the value estimates of states directly from experience. To understand TD Learning we can start with this formula:

$$V(s_t) \leftarrow V(s_t) + \alpha(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)), \tag{2.2}$$

where,

$$V(s_t) = \mathbb{E}[G_t | S_t = s_t] \tag{2.3}$$

This formula iteratively updates the value function $V(s_t)$ of a state s_t, based on the observed reward r_{t+1} for transitioning to the next state and the estimated value $V(s_{t+1})$ of that subsequent state. The update is modulated by the learning rate α and the discount factor γ, facilitating the agent's learning about future rewards and enabling strategy adjustments to optimize long-term gains. In this context, the value function $V(s_t)$ is understood as the expected return $E[G_t|S_t = s_t]$, where G_t is conditioned on the random variable S_t being in a specific state s_t, which is the expected cumulative discounted reward starting from state s_t. The application of TD Learning in this study is instrumental for computing rewards that inform the agent's decisions in navigating the Super Mario environment. By leveraging the TD target to compute rewards, the agent is equipped to assess the immediate and future benefits of its actions, helping it to learn the environment that balances exploration with strategic planning. This approach is not only relevant for enhancing the agent's performance in complex video game scenarios but also sets the groundwork for understanding actor-critic methods, where the concept of advantage calculation plays a critical role in optimizing the policy and value functions.

2.3 Actor Critic

Within the realm of RL, methodologies can be broadly categorized into value-based and policy-based methods, with actor-critic methods emerging as a hybrid approach that combines the strengths of both. Value-based methods focus on evaluating

the value of each state or state-action pair, aiming to maximize the expected reward. Conversely, policy-based methods optimize a parameterized policy directly. This policy, represented as a parameterized function, determines the probability distribution over actions for each state, aiming to maximize rewards by adjusting its parameters based on learning outcomes. Actor-critic methods leverage the advantages of both approaches by employing two models: the actor, which adjusts the parameterized policy towards actions that are expected to yield higher rewards, and the critic, which evaluates the current policy by estimating the value function for states or state-action pairs.

In the actor-critic framework, the advantage function plays a pivotal role by quantifying the relative benefit of taking a specific action a in a given state s over the average action. The advantage function is expressed as:

$$A(s_t, a_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{2.4}$$

Here, $V(s_t)$ and $V(s_{t+1})$ represent the value function estimates of the current state s_t and the next state s_{t+1}, respectively, while r_{t+1} is the immediate reward received after taking action a_t at state s_t. This formulation directly incorporates the specific action a_t, thereby addressing the relative merit of this action compared to the typical outcome expected from the current policy.

The advantage calculation thus measures the difference between the expected return if action a_t is taken at state s_t and the value predicted by the critic for just being in state s_t, focusing on actions that provide higher-than-expected returns. This is critical for updating the policy in a way that pushes the actor to prefer actions that yield better outcomes.

Updates to the actor model are then guided by the following actor loss function:

$$L_{\text{actor}} = -(\log(\pi(a_t|s_t)) \cdot A(s_t, a_t)) \tag{2.5}$$

Here, $\pi_\theta(s, a)$ represents the policy function, parameterized by θ, which dictates the probability of taking action a in state s. This loss function integrates the logarithm of the probability of selecting action a_t given state s_t—as determined by the actor's policy—weighted by the advantage $A(s_t, a_t)$. By doing so, it aligns with the policy gradient method, which encourages the policy to increase the likelihood of actions that lead to higher advantage scores, thus steering the agent towards more rewarding behaviors.

Concurrently, the critic model is updated through the critic loss function,

$$L_{\text{critic}} = (A(s_t, a_t))^2 \tag{2.6}$$

Emphasizing the squared advantage to minimize the difference between the expected and actual returns. This approach reflects the foundational principles of value-based methods, where the goal is to refine the value function estimate to accurately predict future rewards.

The important concept to consider is the actor loss through the policy gradient theorem,

$$\nabla_\theta J(\theta) = \mathbb{E}\pi\theta[\nabla_\theta \log \pi_\theta(s, a) A^{\pi_\theta}(s, a)] \tag{2.7}$$

The policy parameters θ influence both the selection of actions and their evaluation through the advantage function $A^{\pi_\theta}(s, a)$. This function measures the relative benefit of choosing action a over other possible actions given state s, based on the current policy π_θ.

The objective function $J(\theta)$, which needs to be maximized, represents the expected return when following policy π_θ. It quantifies the overall effectiveness of a policy parameterized by θ. The gradient $\nabla_\theta J(\theta)$ of this function concerning the policy parameters θ is calculated by averaging the product of the logarithm of the policy's probability density and the advantage function over all states and actions as determined by the current policy π_θ.

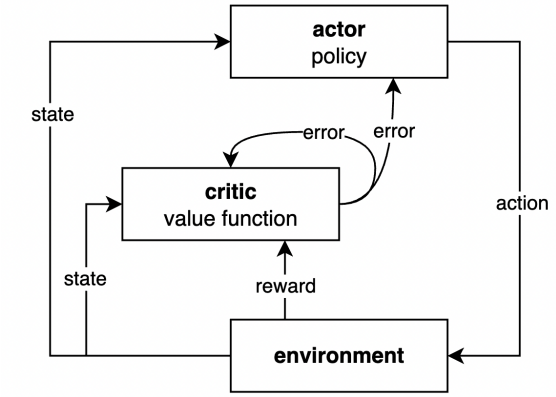

Figure 2.1: Actor-Critic Overview: The process above loops continuously as the agent operates (O'Grady (2024)).

As depicted in Figure 2.1, the actor, dictated by its policy, determines the course of action given the state received from the environment. Subsequently, the environment responds with a new state and an associated reward. This feedback is assessed by the critic through a value function, which evaluates the action's effectiveness based on the received reward and computes an error signal. This signal serves as a learning cue for the actor, guiding policy adjustments aimed at maximizing future rewards. The process, marked by a continual loop of interaction, learning, and adaptation, aims for convergence to an optimal policy, where the actor's

decision-making is improved, and the critic's value predictions become increasingly accurate, embodying the essence of the actor-critic methodology in RL.

In this study, the implementation of actor-critic methods within discrete spaces, such as the Super Mario game environment, showcases the efficacy of combining direct policy optimization with value estimation to navigate complex, dynamic challenges in Super Mario. By incorporating the advantage function into the actor and critic updates, the methodology fosters a nuanced balance between exploring new strategies and exploiting known rewards, underscoring the versatility and robustness of actor-critic approaches in mastering intricate video game tasks. The subsequent section delves into deep learning techniques, which further enhance the actor-critic framework, offering sophisticated tools for representing and solving the complex states encountered in video games.

2.4 Deep Learning

Integrating deep learning into actor-critic methods for Reinforcement Learning bridges the gap between the theoretical framework of RL and practical, high-dimensional applications such as video games. Deep learning models, with their capacity for feature extraction and pattern recognition, are adept at interpreting the complex, noisy inputs of a game environment and making decisions that mirror human-like intelligence.

In the context of this study, deep learning, specifically through the use of neural networks, is important for effectively modeling the policy and value functions within the actor-critic framework. The policy network, which determines the actions taken by the agent, and the critic network, which evaluates these actions, are constructed using neural networks. Deep learning's nonlinearity, provided by activation functions such as the Rectified Linear Unit (ReLU),

$$f(x) = \max(0, x) \tag{2.8}$$

ReLU is particularly advantageous due to its simplicity and efficiency in backpropagation, mitigating the vanishing gradient problem often encountered in deep networks.

Similarly, the softmax function

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for i = 1, ..., K} \tag{2.9}$$

The softmax function takes as input a vector $\mathbf{z}$ of K real numbers and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. It is used in the policy output layer to convert the logits, or the non-normalized predictions from the network, into a probability distribution over available actions. This distribution is essential for the exploration-exploitation trade-off, enabling the agent to stochastically select actions while still favoring those with higher predicted values.

The two architectural choices for actor-critic networks: are separate and shared (two-headed) networks. The separate architecture dedicates distinct networks for the actor and critic, allowing each to specialize in its task but at the cost of increased computational resources and potential redundancy in learning similar features twice. On the other hand, the shared network architecture uses a common set of layers for feature extraction, branching off into two heads for policy and value function outputs. This shared approach ensures that both actor and critic benefit from the same learned representations, which can lead to more cohesive policy and value estimations. However, in a complex space like Super Mario, some features and errors are best to not be shared, to minimize learning noise.

For the Super Mario game environment tackled in this study, the decision between separate and shared architectures is not trivial. It involves considering factors such as computational resources, the complexity of the game environment, and the potential benefits of specialized versus shared learning processes. This trade-off will be further explored in this research.

2.5 Hyperparameters

2.5.1 Optimization Technique

In Safe RL, optimization techniques are pivotal for fine-tuning the parameters of both actor and critic networks to minimize loss functions effectively, thus enhancing the agent's performance. The Adam optimizer is utilized in this study, following the update rule

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.10}$$

Where θ_t represents the parameters at time t, η is the learning rate, $\hat{m}_t$ is the first-moment estimate, $\hat{v}_t$ is the second-moment estimate, and ϵ is a small scalar used to prevent division by zero.

The Adam optimizer, a widely adopted algorithm in machine learning, is favored for its adaptive learning rate capabilities, which allow for larger updates for infrequent parameters and smaller updates for frequent ones. This adaptability is crucial in the context of Safe RL, where the learning algorithm must navigate complex, high-dimensional spaces with efficiency and precision.

2.5.2 Exploration vs. Exploitation

In the realm of RL, the delicate balance between exploration and exploitation is pivotal to an agent's learning and decision-making process.

The actions within this study are sampled from the policy π_θ parameterized by θ, given the current state s_t. The policy outputs a probability distribution over all possible actions, which quantifies the likelihood of each action being optimal given the state s_t. This distribution is formulated as follows:

$$a_t \sim \pi_\theta(s_t) \tag{2.11}$$

This probabilistic approach to action selection inherently incorporates both exploration and exploitation; actions with higher estimated rewards are more likely to be chosen, yet there is always a chance of selecting less-favored actions, allowing for exploration.

During the initial phases, methods such as epsilon-greedy were tested to increase exploration. In epsilon-greedy, the agent predominantly exploits the best-known action, but with probability ϵ, it will explore random actions instead. However, it was found that directly sampling from the policy distribution provided a sufficient balance, implicitly promoting exploration without the need for additional mechanisms such as the epsilon parameter.

Entropy maximization was also considered, which encourages the policy to be more uncertain and thus explore more diverse actions. High entropy in the policy distribution implies a more uniform distribution of action probabilities, which can be beneficial in highly stochastic environments where the agent requires a broader experience to learn effectively. However, this approach interfered with later implemented Safe aspects of RL as we would want to pursue safe paths and explore controllably.

For this study, the final approach settled on sampling actions from the policy distribution, which was deemed adequate for the complexity of the Super Mario environment. The next section will discuss the implementation of Safe Reinforcement Learning heuristics, which are integral to ensuring that the agent's exploration does not lead to unsafe or undesired outcomes.

2.6 Safe Reinforcement Learning

Safe Reinforcement Learning (Safe RL) introduces the concept of incorporating safety within the RL paradigm, especially when the agent interacts with an environment where certain states or actions can lead to negative consequences. This study employs heuristic rules to ensure the RL agent's safety within the Super Mario game. The heuristics serve as predefined rules or conditions that trigger specific actions when certain criteria are met, acting as a safeguard against potentially dangerous decisions that could arise from the agent's exploration.

In the context of Super Mario, state types can be broadly categorized into safe, dangerous, and fatal. Safe states are those in which Mario can navigate without immediate risk of losing a life; these typically involve stable ground. Dangerous states are characterized by the presence of potential threats, such as nearby enemies, or pits, where missteps could lead to death, often allowing for quick corrective action. Fatal states, on the other hand, are scenarios where Mario encounters an unavoidable danger that results in losing a life, such as falling into a pit or directly colliding with an enemy.

The code for this study, therefore, integrates these heuristics by scanning the environment for predefined danger cues, such as proximity to enemies or harmful obstacles, and therefore identifying danger states. These heuristics utilize a template-matching algorithm, a method in computer vision that identifies and locates a template image within a larger image. The process compares the template to all regions in the larger image, searching for matches using pixel intensity patterns. When a match that surpasses a specified threshold of similarity is found, signaling imminent danger, the system initiates a 'hard action.' This action, predetermined and deemed safe, such as jumping away or shifting direction, supersedes the policy-based action the agent would typically execute. This safeguard allows the agent to explore and learn the dynamics of the game environment with a reduced likelihood of entering into states that could lead to unsafe outcomes or failures.

By using heuristic safety rules, the agent is equipped with an immediate response to danger that supplements the learning process. This approach enables the agent to explore and exploit the game environment more safely, reducing the likelihood of engaging in actions that would lead to the termination of an episode due to unsafe interactions.

3 Methodology

3.1 Problem Statement and Hypotheses

Continuing from the established importance of Reinforcement Learning in dynamic game environments, we now narrow our focus to a specific problem within the Super Mario game environment. The challenge we address here is the development of an RL agent that not only navigates the environment effectively but does so with an emphasis on safety.

To formalize this, we are tasked with constructing an agent that employs an actor-critic algorithm where the 'actor' is responsible for selecting actions, while the 'critic' assesses these actions based on a reward signal. The uniqueness of our approach lies in the introduction of a safety mechanism: the agent is designed to switch to a heuristic solution when it encounters imminent danger. This dual strategy aims to mitigate fatalities caused by common in-game hazards, such as the infamous Goombas, while also ensuring that the agent remains capable of learning and exploration.

We aim to investigate whether such an integration can effectively reduce agent fatalities without compromising the efficiency of learning and exploration. We will further delve into the methodology and empirical analysis to test the stated hypothesis.

3.2 Experimental Design

Transitioning from the theoretical framework to practical implementation, our experimental design is tailored to test the hypotheses outlined in the preceding section. The Super Mario game environment serves as the experimental playground wherein the intricacies of the proposed RL agent are examined. This environment is a simulation of the original Super Mario Bros. game by Miyamoto and Tezuka (1985), presenting a two-dimensional grid where the agent, Mario, must navigate through levels filled with platforms, obstacles, and enemies, passing through the challenges of the game.

Within this environment, the RL agent's task is to move from the left side of the level to the right, effectively reaching the goal post, while maximizing the cumulative reward. To assess the agent's performance comprehensively, we employ multiple metrics for success:

- Survival Rate: The proportion of trials where the agent avoids fatal encounters.
- Cumulative Reward: The average total reward obtained across trials, accounting for progress, enemy defeats, and coin collection.

These metrics serve as indicators of not just the agent's ability to survive but also its efficiency and effectiveness in navigating the environment.

Before the agent can interact with the environment, preprocessing steps are necessary to transform the game data into a format conducive to learning. This includes:

- State Simplification: SuperMarioBros-v2 is the version of Super Mario that was used. It has the raw pixel data which reduces the amount of information to process.
- SkipFrame: Designed to reduce the temporal resolution of the environment by returning only every 4-th frame. Helps in reducing the computational load.
- GrayScaleObservation: Reducing the observation space from three color channels (RGB) to a single grayscale channel, significantly lowers the dimensionality of the input data.
- ResizeObservation: Transforming the observations into smaller images, decreases the amount of information the model has to process.
- FrameStack: Observations from several consecutive frames are stacked together to form the input to the model. This technique allows the agent to perceive motion and make more informed decisions based on the recent sequence of observations.

Figure 3.1 shows the resulting state after preprocessing. It is crucial to preprocess as it allows the agent to learn more efficiently by focusing on the game aspects most relevant to the action selection.

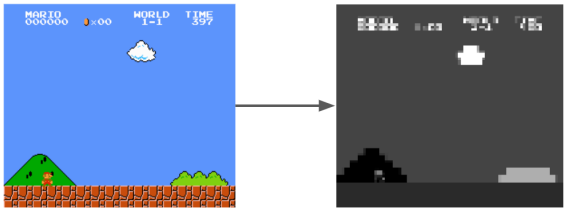

Figure 3.1: Super Mario Bros States: before vs. after the preprocessing.

Within Super Mario, the action space is discrete, meaning the agent's possible actions at any given state are finite and distinct. This is a critical aspect as it dictates the decision-making process and the complexity of the policy the agent must learn. The RL agent interacts with the Super Mario environment via a set of RIGHT_ONLY predefined actions such as going right, jumping, jumping right, and no action. The action space's discreteness allows for a clear evaluation of each action's consequences, essential for both the Actor and Critic components in our Actor-Critic framework. The Actor is responsible for choosing actions, whereas the Critic evaluates their potential based on the current policy.

To effectively train the Actor-Critic model, we calculate the loss for both the Actor and Critic components. The Actor's loss (2.5) is formulated to measure the difference between the expected reward and the obtained reward, as formerly explained. The Critic's loss (2.6), on the other hand, assesses the temporal difference error, reflecting the discrepancy between the predicted and the actual rewards following an action.

The 'Advantage' in the Actor loss represents the difference between the reward of the action taken and the average reward for that state, pushing the Actor to favor actions that perform better than average.

The default reward function $R(s, a)$ is a composite of various elements that reflect the agent's performance:

- r_{position}: This component rewards the agent for moving towards the goal.
- r_{clock}: This penalizes the agent for taking too much time, incentivizing efficient play.
- $r_{\text{fatal state}}$: A penalty is given for dying.

When the agent was training without the use of heuristics, it would resort to instantly killing itself by the hand of an enemy to avoid the time penalty. To counteract this behavior, a modification was introduced to the reward function, imposing a more significant penalty for fatal states by an enemy (goomba, pit). This adjustment aimed to discourage the agent from seeking a fatal state as a viable strategy, thereby promoting longer play durations and exploration. This is crucial as Mario is more favored to avoid fatal states than running out of time.

- r_{enemy}: Penalty is given when the agent dies by the hand of an enemy.

This reward function is pivotal to the RL process as it directly influences the policy gradient and shapes the learning trajectory of the agent. It is the compass by which the agent navigates the landscape of the game environment, learning to prioritize actions that align with the overarching goal of avoiding progressing while avoiding fatal states.

The experimental design is constructed to facilitate an evaluation of the Actor-Critic agent with integrated heuristic safety mechanisms within the rich and varied Super Mario game environment. In the next section, we will explore the implementation of the Actor-Critic framework and the heuristic safety measures integration.

3.3 Implementation Details

3.3.1 Neural Network Architecture

The neural network architecture for both the Actor and Critic components is designed to process the state inputs from the game and output actionable insights. Here's a practical overview:

- Input Layer: Accepts a flattened array of the game state.
- Hidden Layers: Comprises several fully connected layers. The architecture includes two hidden layers with ReLU activation to introduce non-linearity and one feature flattening linear layer.
- Output Layer (Actor): Produces a probability distribution over possible actions, with a softmax activation function to ensure the output values sum up to 1, representing probabilities.
- Output Layer (Critic): Outputs a single value representing the estimated value of the current state.

The number and the type of layers do not change between Separate and Shared Architecture (see Appendix A.1).

3.3.2 Heuristics Integration

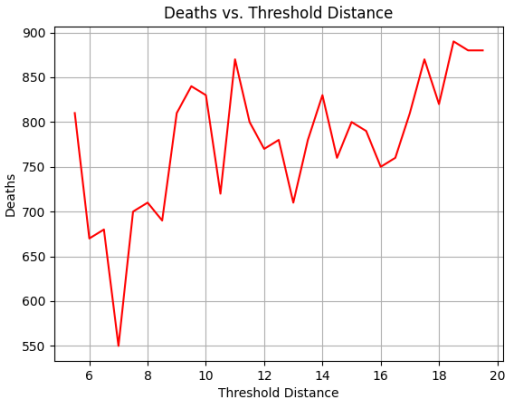

Figure 3.2: Grid Search for the Distance to the Danger Threshold.

The integration of heuristic safety mechanisms plays a pivotal role in the model's ability to navigate the game environment safely. The integration is based on a sliding window of the image of danger across the state image and when the threshold of them being matched is surpassed the area is indicated to be dangerous. For this experiment, we used only one enemy, goomba, as it seemed sufficient to judge the efficiency of our approach and to reduce the complexity. When the distance between the agent and the danger area is below the threshold, the Actor switches to heuristic action (Jump Right) and therefore avoids the danger. To correctly estimate the distance threshold, a grid search was conducted (see Figure 3.2). 1000 episodes for each distance was used to see the number of fatal states, the 7.5 distance seemed to produce the best result. These heuristics are designed to directly influence the $r_{\text{fatal state}}$ component of the reward function. Pseudocode 3.1 illustrating the integration of a heuristic for enemy avoidance:

Algorithm 3.1 Heuristic Integration for Environment Navigation

$\pi \leftarrow$ Model(current state) {get the categorical distribution from the actor model for the current state}
frame $\leftarrow$ ExtractFrame(current state) {preprocessing steps described in section 3.2}
danger $\leftarrow$ DetectDanger(frame) {based on the danger distance, see section 3.3.2}
action $\leftarrow$ SampleAction(π) {see equation 2.11}
if danger $\leq$ threshold **then**
  PerformAction(predefined action)
else
  PerformAction(action)
end if
reward $\leftarrow$ ReceiveReward() {based on the reward function described in section 3.2}
UpdatePolicy(π, action, reward, next state) {based on the fig. 2.1}

This approach ensures that the RL agent not only pursues the objective of level completion but does so with an emphasis on survival, balancing aggressive progression with cautious tactics when necessary.

3.4 Training Process

The training process involves 20.000 iterative episodes where the agent interacts with the game environment, executing actions based on its current policy and receiving feedback in the form of rewards. These rewards are critical for adjusting the agent's policy to improve its performance over time. The steps include:

- Initialization: The agent, composed of both Actor and Critic components, starts with a randomly initialized policy.
- Episode Execution: Each episode represents a full game playthrough, where the agent makes action decisions at each timestep.
- Reward Accumulation: After executing an action, the agent receives a reward signal based on the outcome. It is designed to reflect the agent's performance.
- Policy Update: The Actor and Critic networks are updated to refine the agent's policy. The Actor learns to choose better actions, while the Critic estimates the value of state-action pairs more accurately.
- Repeat: Steps 2-4 repeat across 20.000 episodes, with the agent improving its ability to navigate the game successfully.

Here are the additional hyperparameters that were used for all the trials of this research:

- Learning_rate: 0.0001. It was fine-tuned by observing the dynamics of the agent's learning and estimation.
- Discount Factor: 0.99. The aim was to prioritize future rewards and 0.99 was observed to be a good choice for this.

fatal state within the game serves as a significant learning signal. It directly affects the $r_{\text{fatal state}}$ component of the reward function, imposing a penalty that encourages the agent to develop strategies to avoid fatal mistakes. The inclusion of the fatal state as a metric necessitates a nuanced approach to learning, where the agent must weigh the benefits of aggressive exploration against the potential risks. This dynamic is critical for adapting the agent's behavior to prioritize survival alongside task completion.

The agent's performance is continuously monitored using a metric logger, focusing on critical indicators such as the frequency of fatal states, cumulative rewards, episode length, and loss. This evaluation framework aligns with the actor-critic methodology, emphasizing the importance of reducing fatal state penalties and enhancing cumulative rewards as indicators of learning and adaptation.

This comprehensive training process, based on an adjusted reward signal and a balance between safe exploration and exploitation, enables the RL agent to progressively improve its performance. By placing a significant emphasis on learning from the fatal state as a metric, the agent develops a nuanced understanding of the game environment, leading to more sophisticated decision-making and enhanced adaptability.

4 Results

This section aims to dissect the results obtained from various training phases and configurations, providing a comprehensive analysis of the agent's learning behavior, adaptation strategies, and performance metrics. The examination of various performance metrics such as mean episode length, cumulative reward, mean loss, and death rate over episodes offers insights into the efficacy of the implemented reinforcement learning algorithms. For instance, a longer mean episode length is generally preferable, it suggests that the agent is capable of sustaining gameplay for extended periods, thereby indicating better learning and adaptation to the environment. Or, the 'deaths over episodes' graph provides a direct measure of the agent's progression towards safer operational strategies. A decreasing

trend in this graph would illustrate an improvement in the agent's ability to evade or manage life-threatening situations effectively, thus learning to maximize survival through enhanced policy adjustments. This trend is aligning with the goals of Safe Reinforcement Learning to balance exploration with survival.

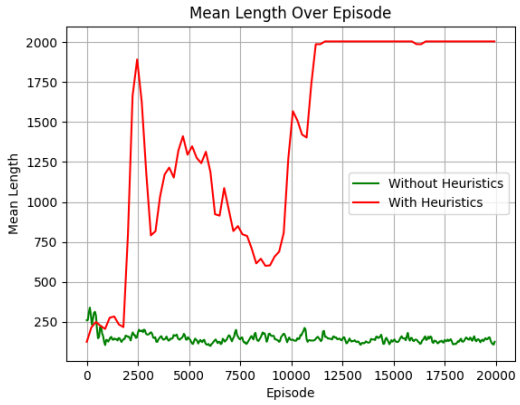

Figure 4.1: Comparison graph of mean episode length over 20,000 episodes between two RL agents: one utilizing heuristics (shown in red) and one without heuristics (shown in green).

This graph (Figure 4.1) compares the mean duration of an episode of two RL agents—one with heuristic safety measures and one without. The y-axis represents the mean length, indicative of the mean duration of an episode. The x-axis shows the episode count over the agent's training period, which spans 20,000 episodes. From the graph, it is evident that the agent with heuristics outperforms the one without heuristics by a significant margin. The agent without heuristics maintains a relatively low and stable mean length, suggesting it does not improve much over time. The observed performance was that the RL agent tended to exploit a loophole in the reward system by moving the fatal states on purpose in-game suicide, as evidenced by the mean length over episodes graph convergence (see green line in Figure 4.1). This behavior was attributed to the agent's discovery that death by enemy hands avoided the time penalty associated with the reward function, inadvertently incentivizing the premature termination of the game.

In contrast, the agent with heuristics shows a dramatic increase in mean length after approximately 8,000 episodes (see red line in Figure 4.1), maintaining this high performance for the rest of the training period. This suggests that after a certain point in its training, the heuristic-enabled agent learned to navigate the environment much more effectively, possibly avoiding dangers that would prematurely end the episode.

The spikes and dips in the mean length of the agent with heuristics may indicate episodes where the agent encountered particularly challenging sections of the game or where the heuristic rules may have had varying levels of impact on its performance. The overall trend, however, clearly demonstrates the benefit of incorporating heuristic safety measures into the RL training process.

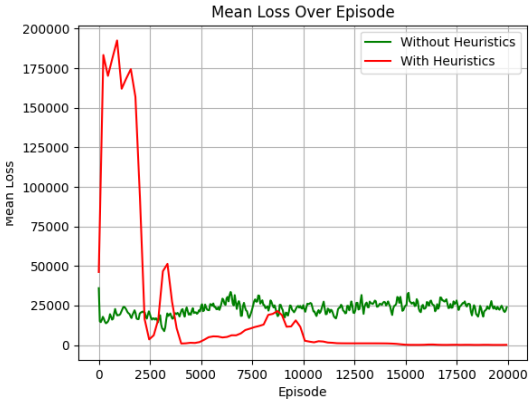

Figure 4.2: Comparison graph of mean episode loss over 20,000 episodes between two RL agents: one utilizing heuristics (shown in red) and one without heuristics (shown in green).

The graph (see Figure 4.2) shows the mean loss of two Reinforcement Learning agents—with and without heuristic safety measures—over 20,000 episodes. The y-axis indicates the mean loss, which is a quantitative measure of how far off the agent's policy is from the actual outcomes. The x-axis is the same as for the Figure 4.2.

Without heuristics, the agent experiences low loss values, and as training progresses, there are no signs of divergence (Figure 4.2). The agent without heuristics shows a stable loss, indicating preferred action dominance over time, but with considerable fluctuation. This variability could stem from the agent's trial-and-error learning process without the guidance of safety heuristics.

On the other hand, the agent with heuristics demonstrates a more dramatic initial reduction in loss, followed by a stabilization at a lower level of mean loss compared to the agent without heuristics. This rapid decrease suggests that the heuristics provide valuable guidance that helps the agent more quickly converge to a better policy by avoiding costly mistakes. The relatively stable and low loss after around 7,500 episodes indicates the agent with heuristics consistently applies what it has learned to make more accurate decisions.

Overall, the graph suggests that incorporating heuristics not only improves safety but also contributes to a more stable and efficient learning progression, as evidenced by a lower and more consistent mean loss over time.

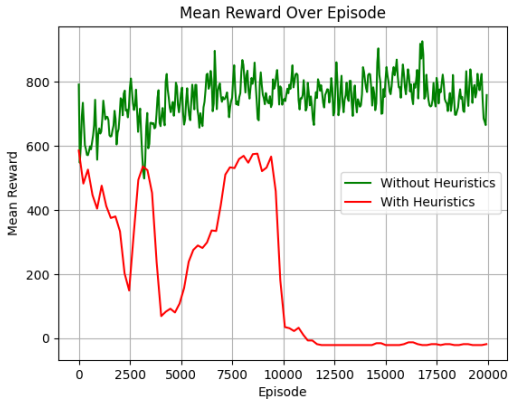

Figure 4.3: Comparison graph of mean episode reward over 20,000 episodes between two RL agents: one utilizing heuristics (shown in red) and one without heuristics (shown in green).

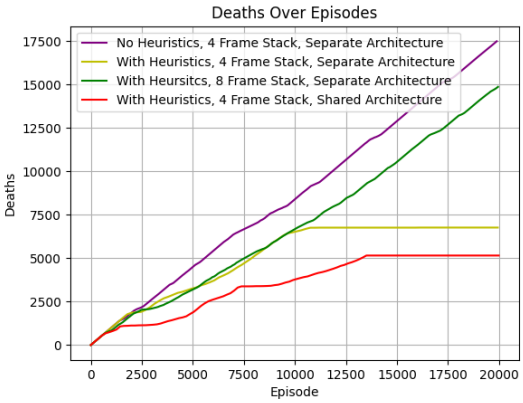

Figure 4.4: Comparison graph of deaths over 20,000 episodes between four RL agents: one with no Heuristics, 4 Frame Stack, and Separate Architecture, one with Heuristics, 4 Frame Stack, and Separate Architecture, one with Heuristics, 8 Frame Stack, and Separate Architecture, and one with Heuristics, 4 Frame Stack, and Shared Architecture

The graph (Figure 4.3) compares the average rewards earned per episode by two agents: one with heuristic safety measures and one without, over the course of 20,000 episodes in the game environment. The y-axis measures the mean reward.

Observing the agent without heuristics, there is a notable variability in the mean reward, with a possible general upward trend. This suggests that while the agent is possibly learning and improving, its performance is still inconsistent. The fluctuations may indicate that the agent is exploring various strategies and learning from a wide range of experiences, including mistakes.

However, the agent with heuristics shows a significantly different pattern. Initially, the mean reward is high, which could indicate that the action selection probability distribution is even and a lot of exploring is happening, therefore causing the agent to be overly exploring, potentially riskier situations. However, there is a sharp decrease to a stable minimum mean reward at around 10.000 episodes into the training. This pattern suggests that initially explores all sorts of outcomes, thus earning a high reward, but as training progresses, these safety measures could be limiting the agent's ability to explore and capitalize on lower-reward survival behavior, resulting in a plateau of lower mean reward.

In conclusion, while the inclusion of heuristics appears to lead to safer and more consistent performance initially, it may also inhibit exploration and the attainment of higher rewards in the later stages of training. This introduces an interesting dynamic between the safety and reward maximization objectives in RL.

The "Deaths Over Episodes" graph (see Figure 4.4), when analyzed comprehensively, reveals insightful trends about the learning process and survivability of the RL agent with different configurations. In the context of the study, the metric of deaths per episode can be seen as the reflection of the implemented strategies. This metric not only reflects the agent's ability to survive longer by making safer decisions but also serves as a crucial indicator of the trade-offs between pursuing higher rewards and maintaining safety. This graph provides the most crucial metric for this thesis, as the Safe aspect is the priority in the conducted research.

- Without Heuristics, 4 Frame Stack, Separate Architecture: Shows a continuous linear increase in deaths, suggesting that the agent fails to learn from its mistakes or that the complexity of the game exceeds the agent's capacity to learn survival strategies without additional guidance.

- With Heuristics, 4 Frame Stack, Separate Architecture: The trend indicates an initial sharp increase in deaths, which then plateaus around the 10000th episode. This early plateau signifies the effectiveness of heuristics in aiding the agent to identify and avoid risks, thus stabilizing its performance quicker than the other configurations.

- With Heuristics, 8 Frame Stack, Separate Architecture: A steady increase in deaths is observed, though less steep than without heuristics. The 8-frame stack may provide more information for decision-making but does not seem to offer the same level of improvement in survival as the 4-frame stack with heuristics.

- With Heuristics, 4 Frame Stack, Shared Architecture: This configuration shows a plateauing of deaths and a less steep convergence compared to separate architecture, but only after about 12500 episodes. Despite eventually reaching stability, the agent takes longer to adapt compared to a separate architecture.

In conclusion, the use of heuristics significantly impacts the agent's learning curve and survival rate, the use of a shared architecture slows down learning and an increase in the frame stack doesn't improve our heuristics approach.

5 Conclusions

5.1 Discussion

The most effective configuration, which pairs heuristics with a 4-frame stack in a separate architecture, illustrates a critical finding: heuristic rules can significantly expedite the learning process by enabling the agent to identify and avoid risks sooner, which in this case happens around the 1000th episode mark.

The early plateau in this configuration suggests that the agent quickly integrates the feedback from its interactions with the environment, adjusted by heuristic-driven safety measures. This rapid learning curve is beneficial for training agents in complex domains where early proficiency is desirable, reducing the computational cost and time associated with training.

On the other hand, it can be argued that even though a shared architecture, demonstrates a delayed response to safety, a lower cumulative death count before plateauing is a positive sign. Delayed convergence could be due to the shared features that heavily complement and at the same time badly influence the actor and the critic. So a trade-off of convergence time and fewer cumulative deaths can be seen here and therefore further explored.

In general, the study's core revelation—that heuristic safety measures markedly improved the safety of RL agents navigating the environment—resonates with contemporary RL research. This aligns with the growing consensus that heuristic approaches can bridge the gap between generic RL algorithms and their application in specific, real-world scenarios, emphasizing safety.

Whereas, the observed fluctuations in mean reward and length across episodes highlight a critical challenge in RL: managing the exploration-exploitation trade-off within stochastic environments. The challenge of fostering effective exploration without resorting to stagnant behaviors required a reevaluation of the exploration strategies. None of the tested methods, i.e. epsilon greedy or adding entropy in the reward function, helped the exploration, prompting a reliance on extended trial runs in the hope of spontaneous exploration driven by the policy distribution. These fluctuations suggest that while heuristics enhance safety by mitigating risks, they may also inadvertently constrain the agent's ability to discover and exploit novel strategies, potentially capping the achievable performance.

The research contributes to the academic discourse by illustrating the nuanced impact of heuristics on learning dynamics within RL. It invites a reevaluation of how heuristic rules are designed and integrated, advocating for a balance that maximizes learning efficiency without overly restricting the agent's ability to explore and adapt.

This study underscores the importance of contextual and environmental understanding in RL applications. It suggests that future research should not only focus on refining heuristic measures but also on exploring the underlying mechanisms through which these measures interact with the agent's learning process.

5.2 Conclusion

The conclusion of this research contains the significant strides made in understanding and enhancing Reinforcement Learning through heuristic safety measures within the Super Mario Bros environment. This study has established that the strategic integration of heuristic rules even though ensures a safer learning trajectory, ends up negatively affecting not only the performance of the RL agent but possibly also in navigating through the game's challenges.

By analyzing the impact of these measures across various performance metrics—episode length, loss, reward, and death rate—the research provides compelling evidence of their heuristics influence. Notably, the improved safety capabilities and reduced fatalities highlight the critical role of domain-specific knowledge, encoded as heuristics, in refining the agent's decision-making processes.

In conclusion, it is important to shed light on the inherent challenges faced by the Actor-Critic model in mastering such a dynamic environment. For instance, the implementation of advanced algorithms like Proximal Policy Optimization (PPO) or Asynchronous Advantage Actor-Critic (A3C) could substantially benefit the agent. PPO, with its objective function designed to prevent large policy updates, ensures a stable and steady improvement, mitigating the risk of destructive large policy swings. A3C, on the other hand, introduces asynchronous update capabilities, enabling multiple agents to explore and learn from different instances of the environment simultaneously. This not only expedites the learning process but also

encourages a diverse set of experiences, preventing the agent from overfitting to a narrow range of states which can often lead to instability in training. Overfitting was usually encountered on the least favorable optimal solutions, e.g. dying immediately. Both PPO and A3C inherently promote experience variety and can be effective in preventing the kind of training instability that might arise from correlated states.

In synthesizing these findings, the research not only contributes valuable insights to the academic discourse on RL but also sets a foundation for future exploration. Through this work, the nuanced interplay between heuristic integration and learning efficiency in RL has been brought to the forefront, marking a significant step forward in the quest to have the full potential of AI in navigating and mastering complex environments.

5.3 Further Research

In the realm of Reinforcement Learning, the intersection with heuristic safety measures within complex environments uncovers pivotal insights yet also reveals several pathways for future research.

Building on the initial implementation of heuristic safety mechanisms, future iterations will aim to improve the precision of danger detection and the agent's responsive actions. This includes refining the heuristic rules and implementing more sophisticated machine learning algorithms that can learn and adapt danger thresholds dynamically. As the agent progresses in its learning journey, these adaptive thresholds will be fine-tuned to provide a balance between the necessary caution and the boldness required to optimize exploration and exploitation strategies within the game.

Given the observed fluctuations in agent performance, further investigation into the mechanisms governing the exploration-exploitation trade-off is an interesting aspect. Research could aim to develop more sophisticated strategies that enable agents to navigate effectively, potentially through adaptive learning rates or context-aware exploration policies.

Moreover, the results from the 8-stack configuration, show that yes, it is unable to converge (see Figure 4.4). However, We can observe that there is a slow decline in deaths, so it could potentially mean that with a significantly bigger input, it takes longer to learn, and may just require many more episodes to converge. It would be interesting to explore how alternating frame stack influences the learning for safety and exploration.

Reflecting on the observation that simplification might yield greater insights than complexity, future research should consider studies that systematically reduce the complexity of RL systems to understand the core principles governing learning and adaptation. Or quite the opposite, further exploring the challenges encountered with the Actor-Critic model that highlights the need for architectural innovations. Future research might explore alternative models or improvements to existing ones, such as A3C or PPO, to better manage the complexities of stochastic environments. If that proves to be inefficient, researching into environments of lower complexity levels, is a good path to start with.

The exploration of these paths could significantly advance the understanding of RL, particularly in how heuristic safety measures can be leveraged to enhance learning efficiency and navigate complex environments. As the field of RL continues to evolve, these research directions offer promising opportunities to address some of the most pressing challenges in developing intelligent, adaptive, and safe autonomous systems.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

Felix Berkenkamp, Matteo Turchetta, et al. Safe model-based reinforcement learning with stability guarantees. *Advances in neural information processing systems*, 30, 2017.

Ruben Nygård Engelsvoll, Anders Gammelsrød, and Bjørn-Inge Støtvig Thoresen. Generating levels and playing super mario bros. with deep reinforcement learning using various techniques for level generation and deep q-networks for playing. Master's thesis, University of Agder, 2020.

Brockman et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Haobo Fu, Weiming Liu, et al. Actor-critic policy optimization in a large-scale imperfect-information game. In *International Conference on Learning Representations*, 2021.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1): 1437–1480, 2015.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, et al. Soft actor-critic algorithms and

applications. *arXiv preprint arXiv:1812.05905*, 2018.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Gabriel Leuenberger and Marco A Wiering. Actor-critic reinforcement learning with neural networks in continuous games. In *ICAART 2018-Proceedings of the 10th International Conference on Agents and Artificial Intelligence*. SciTePress, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Shigeru Miyamoto and Takashi Tezuka. Super mario bros. Nintendo Entertainment System (NES), 1985. Video game.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Jack O'Grady. Rl agent - softmax actor-critic, 2024. URL `https://www.jackogrady.me/reinforcement-learning-solar/rl-agent-softmax-actor-critic`.

Martin Pecka and Tomas Svoboda. Safe exploration techniques for reinforcement learning–an overview. In *Modelling and Simulation for Autonomous Systems: First International Workshop, MESAS 2014, Rome, Italy, May 5-6, 2014, Revised Selected Papers 1*, pages 357–375. Springer, 2014.

Shaker, Togeliu, et al. The 2010 mario ai championship: Level generation track. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(4):332–347, 2011.

Tianye Shu, Jialin Liu, and Georgios N Yannakakis. Experience-driven pcg via reinforcement learning: A super mario bros study. In *2021 IEEE Conference on Games (CoG)*, pages 1–9. IEEE, 2021.

Julian Togelius, Sergey Karakovskiy, Jan Koutník, and Jurgen Schmidhuber. Super mario evolution. In *2009 ieee symposium on computational intelligence and games*, pages 156–161. IEEE, 2009.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations*, 2016.

A Appendix

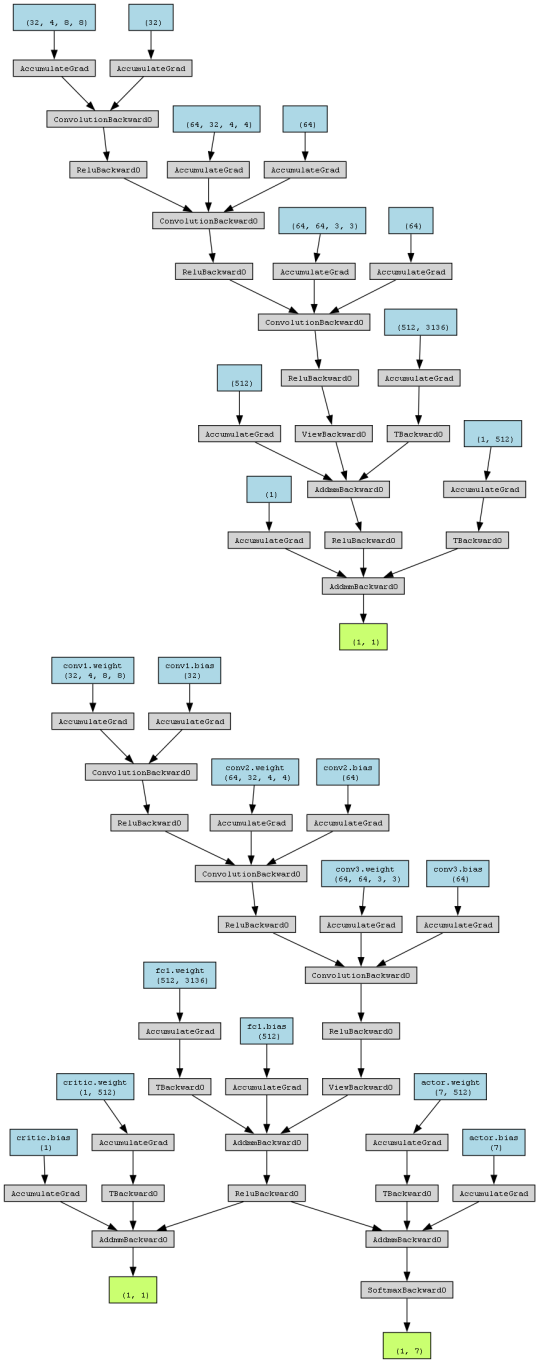

Figure A.1: Neural Network Visualization: a) Shared Architecture; b) Separate Architecture