12-2023

Robust Uncertainty Estimation Framework in Deep Reinforcement Learning for Active SLAM

Bryan Joseph Pedraza
The University of Texas Rio Grande Valley

ROBUST UNCERTAINTY ESTIMATION FRAMEWORK IN DEEP REINFORCEMENT LEARNING FOR ACTIVE SLAM

A Thesis

by

BRYAN J. PEDRAZA

Submitted in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Major Subject: Electrical Engineering

The University of Texas Rio Grande Valley

December 2023

ROBUST UNCERTAINTY ESTIMATION FRAMEWORK IN DEEP REINFORCEMENT LEARNING FOR ACTIVE SLAM

A Thesis
by
BRYAN J. PEDRAZA

COMMITTEE MEMBERS

Dr. Dimah Dera
Chair of Committee

Dr. Hasina Huq
Committee Member

Dr. Wenjie Dong
Committee Member

December 2023

ABSTRACT

Pedraza, Bryan J., Robust Uncertainty Estimation Framework in Deep Reinforcement Learning for Active SLAM. Master of Science (MS), December, 2023, 57 pp., 2 tables, 12 figures, references, 36 titles.

Autonomous mobile robots are essential in various domains such as industry, manufacturing and healthcare. Navigating autonomously and avoiding obstacles are crucial tasks that involve localizing the robot to explore and map unknown environments without prior knowledge. Simultaneous localization and mapping (SLAM) present significant challenges. In this study, we introduce a new approach to address robust navigation and mapping of robot actions using Bayesian Actor-Critic (A2C) reinforcement learning. The A2C framework combines policy-based and value-based learning by dividing the model into two components: (1) the policy model (Actor) determines the actions based on the state, and (2) the value model (Critic) evaluates whether the agent's action depending on its return value from being ahead or behind in an environment/game. This feedback guides the training process in which both models interact and optimize their outputs over time. To achieve robust exploration and collision-free navigation, we develop a Bayesian A2C model that generates robot actions and quantifies the uncertainty associated with these actions. Our approach incorporates Bayesian inference and optimizes the variational posterior distribution over the unknown model parameters using the evidence lower bound (ELBO) objective. We employ a first-order Taylor series approximation to estimate the mean and covariance of the variational distribution when passed through non-linear functions in the A2C model. The propagated covariance estimates the robot's action uncertainty at the output of the actor-network. Experimental results demonstrate the superior robustness of the proposed Bayesian A2C model when exploring environments with high levels of noise compared to deterministic alternatives. Furthermore, the proposed framework has the potential for various applications where robustness and uncertainty quantification are crucial.

DEDICATION

This thesis is dedicated to my parents, Jose and Lilly Pedraza, for their love and continuing support to serve as the foundation of my life and academic journey. Their belief in me and my capabilities has empowered me to pursue my aspirations, and their sacrifices have paved the way for my accomplishments.

I also dedicate this work and express my gratitude to my mentor, Dr. Dimah Dera. Her guidance and motivation have been a constant source of inspiration throughout my academic journey. Her expertise and dedication have motivated me to strive for excellence in my studies.

I extend my appreciation to every individual who played a role in this journey and express my gratitude to each one of them.

ACKNOWLEDGMENTS

I am filled with deep gratitude and appreciation for all those who have guided and supported me throughout my journey of completing this thesis. Without their unwavering support, insightful guidance, and encouragement, this work would not have been possible.

I am immensely thankful to my advisor, Dr. Dimah Dera, for her invaluable guidance, patience, and expertise. Her insightful feedback, knowledge, consistent encouragement, and willingness to dedicate her time and effort to discuss ideas have been instrumental in shaping the direction and quality of this research.

I sincerely appreciate the feedback and valuable suggestions provided by the members of my thesis committee, Dr. Dimah Dera, Dr. Hasina Huq, and Dr. Wenjie Dong, which have greatly enriched the content of this thesis.

I would like to acknowledge the University of Texas at Rio Grande Valley (UTRGV) for providing the resources, facilities, and academic environment necessary for conducting my research. I am also grateful for the financial support provided by the UTRGV Electrical Engineering department and the GAANN Fellowship program, which enabled me to carry out this research and present it in conferences and experiments.

Lastly, I want to express my deepest gratitude to my family and friends for their encouragement, support, unconditional love, and constant belief in me throughout my journey. Your constant support and belief in me have been the driving force behind my achievements and dedication.

In conclusion, this work would not have been possible without the collective efforts, contributions, and support of everyone mentioned above. I am truly grateful for the opportunity to undertake this research and for the guidance and support that has accompanied me throughout this journey. Thank you for being a part of my academic and personal journey.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER I

INTRODUCTION

1.1 Motivation and Problem Statement

Reinforcement learning has been an area of great interest in machine learning for quite some time. Agents have recently shown remarkable performance in learning and outperforming human-level skills [16, 35, 18], which may improve one area of interest known as Simultaneous Localization and Mapping (SLAM). This is where a robot/agent equipped with a LiDAR (Light Imaging Detection and Ranging) and/or camera localizes itself and maps an environment. The approach to have the agent autonomously navigate on its own is known as active SLAM [1, 21]. The robot would need to actively explore, navigate, map an unknown environment, and gather useful data to construct a map of the real-world environment. However, the problem remains in exploring unknown environments, which is challenging due to the agent/robot not having any prior knowledge about the environment.

Deep reinforcement learning (DRL) facilitates learning based on some initial conditions, exploring and navigating unseen environments [34]. Many DRL approaches have been proposed in the literature to solve the active SLAM problem [19]. However, quantifying uncertainty in the robot's actions, especially important for cases where the agent undergoes noisy or corrupted environments, is still missing. This thesis explores how estimating uncertainty in the robots' actions and mapping for robust exploration and navigation in unknown environments (active SLAM problem) may improve the performance and robustness compared to the traditional deterministic models, particularly when the robot goes into noisy environments. This study aims to investigate and develop a framework for handling uncertainty and assess the advantages of incorporating

uncertainty into the system when addressing the active SLAM problem. Numerous techniques have been proposed in previous studies to tackle active SLAM problems, including filter-based methods, graph-based methods, optimization-based methods, and others [5, 12, 31, 34]. Nonetheless, effectively exploring unfamiliar environments remains a daunting task due to the limited prior knowledge and the essential requirement of accurately measuring uncertainty, particularly in highly noisy environments and the impact on robustness in such cases.

1.2 Research Objectives and Contributions

This thesis introduces an innovative Bayesian Actor-Critic (A2C) model that applies deep reinforcement learning for Active Simultaneous Localization and Mapping (SLAM) algorithm. The proposed Bayesian A2C algorithm effectively estimates the actions performed by the robot while also quantifying the uncertainty associated with those actions. By leveraging this uncertainty, the robot can navigate in the correct direction and avoid potential collisions. To achieve this, we employ Bayesian inference and optimize the variational posterior distribution of the A2C model parameters using the Kullback-Leibler (KL) divergence, which minimizes the difference between the approximate and true posterior distributions. The objective function in this optimization problem is known as the evidence lower bound (ELBO) objective function. Building upon the existing uncertainty propagation framework described in [7], we extend it by propagating the mean and covariance of the variational posterior distribution through the Bayesian A2C networks. In order to approximate the mean and covariance after non-linear layers, we employ the first-order Taylor linearization. The resulting propagated covariance at the output of the actor network provides a measure of uncertainty for the predicted action. This quantification of uncertainty enhances the robustness of autonomous robot navigation, particularly when operating in noisy environments. Our experimental results demonstrate that the Bayesian A2C model achieves higher reward values compared to the deterministic A2C model. Furthermore, the Bayesian A2C model maintains consistent performance and high reward values, even when the robot operates in a noisy environment during validation without being trained specifically for that noisy environment.

1.2.1 Contributions

In particular, the research contributions are summarized as follows:

1. Build a novel active SLAM algorithm to efficiently learn to navigate and explore in an unknown and highly noisy or malicious environment.

2. Integrate Bayesian theory in deep reinforcement learning in order to measure the uncertainty of the predicted action using prior and posterior distributions over the Actor and Critic networks' parameters.

3. Improve the model's robustness during exploration and object collision avoidance when undergoing noisy conditions or corrupted unseen areas of the environment, using estimated uncertainty in the predicted actions.

4. Investigates the essential aspects of reliability and dependability in intelligent robotic systems. A well-structured theoretical and algorithmic foundation is developed, which has an influence on the creation of mapping and active sensing algorithms for autonomous robots. These abilities are important in areas such as security, surveillance, and environmental monitoring, where promptly acquiring knowledge about the surrounding terrain, structures, and human presence is crucial.

The scope of this research is to find an answer to the following questions:

1. How can uncertainty in the robot's actions improve the exploration, navigation and mapping of the environment?

2. How can the robot efficiently learn to adapt to sudden changes in the environment using uncertainty in case of noisy or malicious environments?

3. How can uncertainty measurement improve the robustness of the robot's actions and act as a warning signal or failure mode detection mechanism to the agent itself?

1.3 Thesis Organization

This thesis is organized as follows. Chapter II discusses the literature review of deep reinforcement learning and active SLAM algorithms. Then, Chapter III explains the Bayesian Actor-Critic reinforcement learning algorithm. Chapter IV presents the experimental results and the discussion. Finally, Chapter V includes the conclusion and future work.

CHAPTER II

LITERATURE REVIEW

2.1 Deep Reinforcement Learning (DRL)

Deep reinforcement learning (DRL) combines the recent deep neural networks with a framework of reinforcement learning (RL), which refers to goal-oriented algorithms. RL algorithms help software agents learn how to achieve a specific goal; for example, they can maximize the points won in a game over many moves [17, 16]. Although, in the beginning, DRL algorithms will not know exactly what proper actions to do in a given space of an environment (can start from a blank state), the agent learns through trial and error (reinforcement) to optimize to the best policy action in order to reach the desirable end goal. The reward function is used to guide the agent. Developers set up the reward function to determine how much to reward or penalize the agent after doing an action and observe the changes in the environment. Due to this rewarding concept, some researchers consider DRL to be semi-supervised learning (between supervised and unsupervised) because the agent is slightly guided to an appropriate optimal policy action through rewards. However, the agent is learning (optimizing) on its own through trial and error to achieve the goal [28].

The space in an environment can be very huge. For example, an agent learning how to play chess can be challenging because the spacing is so vast in this grid world with a finite set of actions. If the opponent or agent just moves one piece, the space of the environment is different than before. Therefore, an agent must learn to take the best action in order to receive the best possible reward and maximize its chance to win (end goal). Though this is only one example that explains how spacing in environments can be huge, it can be even more in the real-world environment, where objects or humans can shift position dynamically. Because of this complex spacing and

decision-making on what actions to take, deep neural networks were introduced into the agent itself to help compute non-linear complex functions in an environment. RL tries to mimic human thinking in a trial-and-error way, such as human learning how to play tic-tac-toe for the first time [27]. The human plays over and over again. It might lose (penalize), or it might win (reward).

Researchers are consistently working on enhancing active SLAM by introducing new techniques, methods, and solutions. Several RL methodologies, including Q-learning, Deep Q-Networks (DQN), double DQN, dueling double DQN, and Actor-Critic (A2C), along with their variants, have been proposed in the literature [34, 20, 19, 2, 33]. However, there has been a notable absence in the literature on how to integrate the uncertainty associated with predicted actions and its impact on the behavior of robots operating in various noisy environments. Current robotic systems lack robust mechanisms for both exploring the surroundings and effectively managing uncertainty, which is crucial for intelligent decision-making systems.

2.2 Theory of Optimal Experimental Design (TOED): Exploration and Mapping in Environment

A few attempts have been introduced in the literature to quantify the agent/robot's path planning to improve its localization and mapping accuracy. This can be done during the robot's exploration into new areas or exploitation in areas that have already been explored. This is achieved by choosing a form of Theory of Optimal Experimental Design (TOED) function [22, 14], in other words, an optimal criteria method for the robot to follow. A criteria method can lead the robot to decide on where to move and what sensor measurements to take based on what it understands about the environment. This leads to more efficient exploration and mapping of self-localization and its environment. There are different criteria methods, such as T-optimality (T-opt), A-optimality (A-opt), D-optimality (D-opt), E-optimality (E-opt), and Shannon's entropy. However, Shannon's entropy originated from information theory (IT) but is very similar to the optimality methods.

Research papers [20, 6, 23] have discussed and compared different optimal criteria methods that best decide a robot's motion planning. The robot's motion planning is based on its ability to accurately localize itself and its pose while completing a map of the environment. The paper states

that reducing the uncertainty on the map helps the robot better understand its own localization and environment by exploring the areas it has not seen before while also completing an accurate map.

Previous studies have demonstrated that from the covariance matrix (Σ), which represents the SLAM map, the T-opt can capture the average variance, the A-opt captures the harmonic average, the D-opt captures the full dimensions of the map, and the E-opt captures a single minimum or maximum eigenvalue. The results claimed that the D-optimal performed better due to its method of capturing the whole map's representation by using all landmarks in the formula method. For example, if the robot is unsure about an area on the map, the robot may decide to move closer and take measurements from that section to gain more information about the environment. The D-optimality criterion from the TOED function is represented in the Equation 2.1 [19, 23, 2]:

$$D\text{-}opt \triangleq \exp\left(\frac{1}{\ell}\sum_{k=1}^{\ell}\log(\lambda_k)\right) \tag{2.1}$$

The ℓ represents the length of the covariance matrix (Σ). The Equation 2.1 contains the eigenvalues (λ_k), which are correlated to represent the landmarks in the map's representation. As mentioned before, the TOED and IT provide a variety of different methods for calculating different uncertainty metrics on the exploration map in unknown/unseen areas. This is how the map is represented in active SLAM's algorithm [6]. The TOED introduced a logarithmic approach in restricting the optimality criteria to never converge into zero, which became the D-opt function, which is similar to Shannon's entropy. This is prone to better exploration, capturing the global areas where the robot is unsure of or areas where it didn't map before. This function plans trajectories (path plan) to explore outwards around the environment and helps navigate, explore, and build the SLAM map. The metric system should then have an idea of where to explore and a path plan to reduce areas where the robot is unsure. The system should also have the ability to gather landmarks information, self-localization, and map building.

Note that in the agent's position in the metrics method, exploration of unseen areas and

mapping are done in the active SLAM algorithm. This algorithm runs independently from the robot. This means that inside the Robot Operating System (ROS) [13], the SLAM algorithm is an entirely different code/node than the robot itself. Therefore, the robot (node 1) communicates with the active SLAM algorithm (node 2) and vice versa. It communicates in helping build the environment's map and the robot's position during exploration by receiving the robot's LiDAR (or camera) scan measurements through the ROS framework. Another useful software inside the ROS framework will be the Gazebo simulator (robot's environment), where the robot will interact and train repeatedly. Also, the OpenAI gym-gazebo extension [36] was used for both training and testing purposes. On the robot and environmental setup. This open-source library provides numerous preconfigured robots and environments, making them readily available for AI-related tasks. The latest DRL approaches are the artificial intelligence network (Deep-Q Network [16], Actor-Critic [12]) and their variations. These approaches work inside the robot and communicate with the active SLAM algorithm. Therefore, the whole framework learns better policy actions based on maximizing future rewards on a current state and learns actions to explore regions that the robot is unsure of or has not seen. The following sections will further discuss these networks and some of the approaches that previous researchers have done.

2.3 Q-learning in Reinforcement Learning

Q-learning is a reinforcement learning policy that will find the best action given a current state. It chooses this action at random and aims to maximize the reward. Consider an advertisement recommendation system. Usually, when we look up a product online, we get advertisements which will suggest similar products over and over again. Using Q-learning, we can create an Ad recommendation system, which will suggest related products to our previous purchase. The reward will be if the user clicks on the suggested product.

In Q-learning, there are some important terms, such as the state, action, reward, episodes, Q-values, and temporal difference (TD) learning [11]. The state, s, represents the current position of an agent in an environment. The action, a, is the step taken by the agent when it is in a particular state. For every action, the agent will get a positive or negative reward. The episode happens when

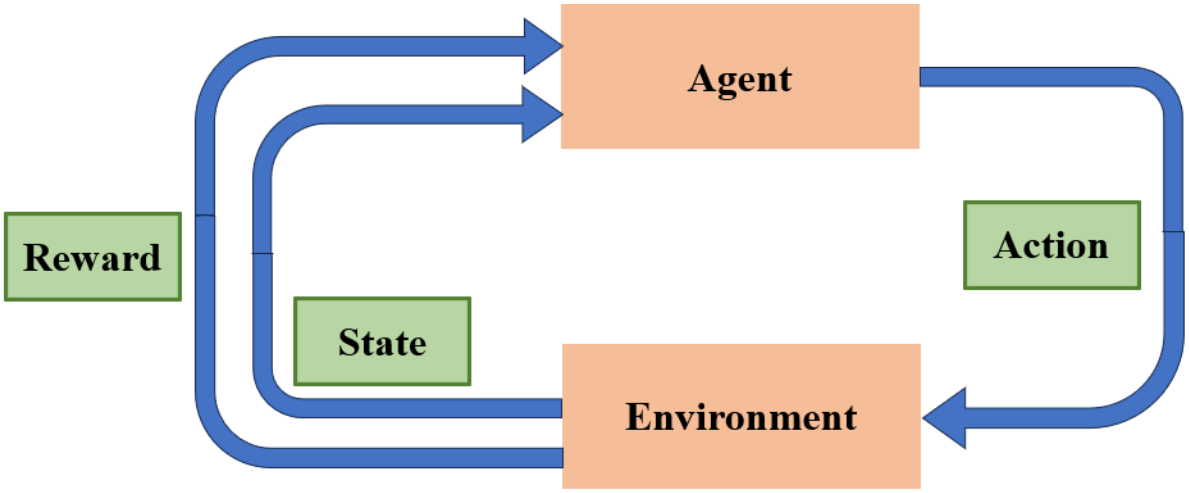

Figure 2.1: A schematic layout of Q-learning.

an agent ends up in a terminating state and can't take a new action. Q-values are used to determine how well an action is taken in a particular state, i.e., $Q(s,a)$. The temporal difference learning is a formula used to find the Q-value by using the value of the current state and action and the next state and action. Figure 2.1 presents a schematic layout that represents Q-learning.

2.3.1 Bellman Equation

The Bellman Equation is the essential building block of reinforcement learning. The equation determines the long-term reward given the state we are in and assumes that we take the best possible action currently and at each subsequent step. The equation is given as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max Q'(s',a') - Q(s,a)\right], \tag{2.2}$$

where (s,a) refer to the current state and action, s',a' refer to the previous state and action, R is the reward, γ is a weighting factor and α is the learning rate.

2.4 Deep Q-Network (DQN) and Reward Function

Deep Q-Network (DQN) uses both Q-learning and deep learning, which contains a deep neural network (DNN) with multiple layers within the agent itself. Along with the loss function, the reward function for DQN helps efficiently optimize the weights within the agent's network and

learn better quality action policy for an environment. Particularly, the Q-function, known as the action-value function, is used to quantify the expected reward by following a policy and taking an action given a state. The DQN learns an optimal Q-function (a policy) to grant the robot the best possible actions to take in order to maximize its expected reward over a number of iterations [34, 19, 25]. The Q-function is seen as a return value by taking a specific action in a given state. The Q-function is represented by $Q(s,a)$, where s is for the state, a is for the action, and Q is for an expected reward value or return value. A few variations of DQN methods have been proposed in the literature, such as standard DQN [16], double DQN (DDQN) [30], rainbow [10], dueling DDQN (D3QN) [32].

Placed and Castellanos approached the active SLAM by model-free DRL, or more specifically, applied a D3QN architecture with Prioritization Experience Replay (PER) adopted from [24] to stabilize and optimize the agent's action policy [19]. The D3QN framework used a mixture of double DQN and dueling DQN and showed better performances and stability over the other Q-networks. The robot is equipped with just a LiDAR sensor and no camera in this approach. The author used a standard squared error loss function on the model and the Q-learning function value, which is used for updating the weights of the D3QN. The agent receives good rewards while it is exploring; thus, guiding the agent to explore, map, and avoid collision is the goal.

On the other side, DDQN and D3QN used the advantaged function that helped the network learn the advantages of taking a particular action over other actions and quantified how much better or worse the action was. The advantaged function is calculated by the differences in reward value received taken by an action and the predicted average value for all possible actions in a given state. The return value (Q-value) is used as part of this function that is subtracted by the average value in the current state. This is written as $A(s,a)$, where both action and state are paired with the advantage. This is also known as the temporal difference (TD) learning [31] used as feedback to quantify how much the current estimate value needs to be adjusted for better outcomes. The optimization problem adjusts the function in a direction to reduce the TD error, thus helping the robot make better predictions about the expected return values and rewards by the action and the state.

The role of the reward function is to guide a robot's actions. The function helps the robot transition from a state of ignorance about its environment and action policy to a state of learning, where it can distinguish between acceptable and unacceptable actions in a given state. Backpropagation in DRL is used in conjunction with the reward function to enable the robot to learn from previous interactions and understand good actions versus bad ones. This approach helps the robot make desirable decisions and take actions that lead to the desirable end goal. Essentially, the reward function acts as a rule book that the agent must abide by, considering the agent has no prior knowledge of the environment, the goal, or the actions. Through DRL and rewards, the robot is guided towards its target goal by penalizing bad actions and rewarding good ones. Therefore, it is imperative to decide the structure of the reward function for an artificial network, as it plays a critical role in guiding the agent to perform optimal actions that lead to achieving specific goals. The approach in [19] is to satisfy the active SLAM cost function $\mathbf{J}$ [6, 14]. This function computes the sum of all free collision trajectories ($\mathbf{T}_i$) and the expected uncertainty of the covariance map ($\mathbf{U}_i$) to determine the best policy class that optimizes the cost function $\mathbf{J}$. Additionally, the approach introduces two other variables, α and β, to represent the constant value for both sums, which are low decimal values. In other words, the approach aims to enable the robot to navigate freely, avoid collisions, and measure the unseen areas around the agent's environment through the SLAM map's representation.

$$\mathbf{J} = \sum_i \alpha \mathbf{U}_i + \sum_i \beta \mathbf{T}_i \tag{2.3}$$

The active SLAM cost function $\mathbf{J}$ is presented as Equation 2.3 consisting of the collision-free trajectory ($\mathbf{T}_i$) and the potential unseen areas in the map ($\mathbf{U}_i$), along with two low-value constant variables, α and β [6].

Placed and Castellans translated the cost function in [6, 14] into a reward function. They trained an agent in DRL to find the best optimal policy and optimize active SLAM function in navigating and exploring [19]. By translating the cost function into a reward function, the agent can be guided to the optimal goal of navigating and exploring. The reward function is very similar to

the cost function in active SLAM but used in different methods and variables. Placed and Castellans proposed a general reward function and demonstrated how collision-free rewards ($\mathbf{R}_f$) and exploring unseen areas ($\mathbf{R}_u$) are individually calculated in the reward function [19]. This is heavily related to Equation 2.3, but in a reward function for the network.

$$\mathbf{R} = \mathbf{R}_u + \mathbf{R}_f \tag{2.4}$$

The reward function in the Equation 2.4 includes a reward when there is no collision, $\mathbf{R}_f$, and a reward when successfully exploring the environment/areas that haven't seen, $\mathbf{R}_u$. Here $\mathbf{R}_f$ is essentially the same as $\mathbf{T}$, and $\mathbf{R}_u$ is the same as $\mathbf{U}$, but wrapped in a reward function. The function penalizes the agent harshly when there's a collision or very little when there's a turn. This is a standard rewarding method in guiding the agent for collision-free navigation seen in previous research [34, 4].

More importantly, the author uniquely computed the $\mathbf{R}_u$ reward for when the agent successfully explored the unseen areas of the environment. From the author's reward function, the f(Σ) is a criteria function used to compute the unexplored areas on the SLAM map and, in this case, the D-optimally criteria from TOED. The author proposed the f(Σ) function or D-opt criterion applied to the denominator of the equation because when the uncertainty of the environment is large, the reward should be small, but when the uncertainty is small (explore unseen regions), the reward should be high. This means that the agent will get positive rewards when it is successful in exploring the environment, reducing the uncertainty. The numerator is a constant task-dependent scale factor wrapped around a tanh(.) function that bounds the reward range from zero to one ([0,1]). This ensured that the $\mathbf{R}_f$ reward was not overshadowed, maintaining a balance between both reward values ($\mathbf{R}_f$ and $\mathbf{R}_u$). The agent optimizes an optimal policy to learn a good strategy in navigation and exploration by the reward function, guiding the agent to achieve this goal in active SLAM.

2.5 Actor-Critic

Although DQNs are useful in DRL by optimizing weight parameters to achieve a better optimal $Q(s,a)$ function and policies to attain a specific goal, the robot will only have a set of discrete actions to take. The actions are determined by the DQN structure in learning to either go forward, turn right, turn left, etc., based on the input state of the environment [16, 30]. This is particularly useful for agents that need to learn the best action to take in a discrete set of environments, such as a board game like checkers, where the player can only take a full step or no step. However, most robots have some type of degree of freedom (DOF) joints like a two-link manipulator, servos, arms, hands, fingers, or in some cases, motors in controlling the speed and angular turn on the wheels. Robotics are all different in what they do and how they are built, but the common similarity between all is that each needs to be able to deeply learn and take action in a continuous or discrete space [12, 15].

The actor-critic (A2C) algorithm is the better choice for robotics and is another method used in DRL where the robot has either a continuous or discrete output space of the actions. Actor-critic is very similar to DQN, but instead of one network model (Q-network), there are two network models. The whole model is split into two; one network is the actor, and the other network is the critic. Both networks are independent of one another and, therefore, have independent weight parameters and compute different outputs. These networks are deep neural networks, and both actor and critic work together to determine the best-desired action. The actor network decides which action should be taken, and the critic network criticizes the actor for every action it decides and informs the actor how good or bad the action was in a given state and how it should be adjusted. The critic's job is to criticize the actor or give it a value score of the state based on evaluating the chosen action and reward, much like a parent criticizing every action in each state. This is done with the TD learning (or advantage function) that is sent to the actor by the critic.

The critic's role is to predict values and the return values for each action paired with the state. All the value placements for action paired with state are done by the critic, and all the actions are predicted by the actor. Because the network is split, the actor can solely focus on actions and

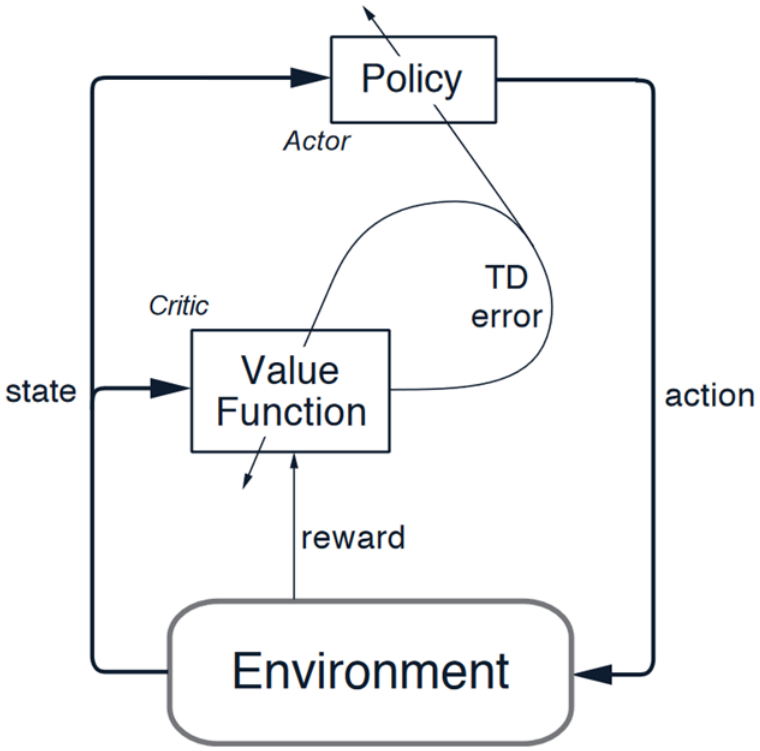

Figure 2.2: The basic structure of the Actor-Critic (A2C) method [26].

freely predict the output of the action with the help of the critic. This is very useful in the robotics aspect in giving the robot learning ability and goals while allowing the robot to maneuver in an open space. Figure 2.2 illustrates the basic structure of an AC2 method [26].

Many different variations of the A2C algorithm have been implemented in the literature, such as asynchronous advantage actor-critic (A3C) [15], and soft actor-critic (SAC) [9] that showed improvement compared to A2C. In a recent research article [33], a hybrid approach used a depth camera and a LiDAR sensor equipped onto the robot instead of just having a LiDAR sensor seen in other research papers. Here the depth-camera image is the input to the DRL network that is composed of a convolution neural network, relational network, max-pooling, and two split independent branches for the actor and critic networks.

Wen *et al.* introduced a unique method for the A2C framework [33]. The paper proposed a network structure of exploration during real-time depth-camera feedback (image frame) on the robot's network model. The authors also proposed a network structure after the convolutional neural

network (CNN) (as a feature extraction technique) that is split into two branches. The actor network and critic network decide on the action and compute a value and TD error of the state environment. However, Wen *et al.* had the actions set in a continuous space for the linear velocity and the angular velocity in order for the robot to be able to learn to maneuver in an action range set [33].

Plenty of research papers developed useful methods for approaching the ongoing active SLAM problem. However, they did not consider uncertainty in the robot's action. The literature failed to approach these issues and even failed to show the model's robustness to noise, nor test the model when the robot undergoes noisy conditions or when images are corrupted in an environment. In [19], Placed and Castellans attempted to solve the active SLAM problem with a DQN, as mentioned before. However, their experiment results were only compared to DQN family methods (DDQN and D3QN). They failed to compare the proposed method with an A2C or present the model's performance in noisy environments. This thesis raises the question of whether A2C performs better or just as well as the DQN family methods in test simulations in solving the active SLAM problem and whether including uncertainty can enhance the robust features when the robot is under disturbed conditions.

Although Wen *et al.* employed an A2C model to solve the active SLAM problem and used complex environments during training to better prepare the robot for the real world, the paper still failed to include uncertainty in exploring and mapping or in the network actions [33]. Moreover, the paper didn't show the robustness of the model under noisy environments. To the best of our knowledge, all exploration methods ultimately failed to use the SLAM map in their algorithm or reward function to guide the robot in exploring unknown areas in the environment. More importantly, no uncertainty was calculated. The authors just created a SLAM map and estimated the position of the robot on the map but didn't use the map in the algorithm for any type of guidance for the robot.

Environments are not always clean, and signal interference happens, especially on machines/robots. Considering these conditions can be very crucial in testing the robustness of these models and can lead to better performance when deployed onto real-world robots. In the real world, there are a lot of signals in the air generated by hundreds of devices, and the uncertainty can

go a long way in measuring how certain the robot is about its own prediction. Uncertainty is an important factor to implement, and a recent study [7] showed that uncertainty can be very helpful to a model's prediction and data distribution of an environment. Including uncertainty may improve the framework's robustness whenever the model is under corrupted/noisy conditions. Uncertainty measurement in the predicted actions may help predict better choices that are closer to the ground truth regardless of an interrupted image.

2.6 Uncertainty on Network

Dera *et al.* proposed a unique approach to computing the uncertainty throughout a DNN using Bayesian theory by estimating the posterior distribution of the unknown parameters [7]. They developed a model called PremiUm-CNNs (Propagating Uncertainty in Convolutional Neural Networks) [7]. This method introduced a normal distribution over the CNN's kernels (sometimes called filters in CNN) as the prior distribution and estimated the posterior distribution over the parameters. That allowed the network to propagate the uncertainty through the network. Bayesian theory is the start of this approach where there is a prior distribution $p(\Omega)$, a likelihood distribution $p(D|\Omega)$, and the posterior distribution $p(\Omega|D)$, which then captured the total knowledge about the hypothesis (Ω) given the evidence (data), hence, Bayesian theorem equation [8].

Using the Bayes rule to estimate the uncertainty is very useful, yet the exact Bayesian inference in DRL with the high dimensional parameter space is intractable (mathematically). We approximate the true posterior distribution of the unknown parameters using the Variational Inference (VI) framework [3]. The idea of VI is to approximate the true unknown posterior distribution $p(\Omega|D)$ with another parametric distribution known as the variation distribution $q(\Omega)$. This is chosen to be the simplest distribution, which is the Gaussian distribution, because with the Gaussian distribution, the whole distribution can be fully characterized by the statistical parameters, I.e., the mean (μ) and the covariance matrix (Σ). We will minimize the Kullback-Leibler (KL) divergence between the variational distribution $q(\Omega)$ and the true posterior distribution $p(\Omega|D)$ [29, 3]. The KL divergence is a known metric that computes the similarity measures between distributions. In other words, we would like to have variational distribution $q(\Omega)$ as close as possible to the true posterior distribution

p(Ω|D). The objective function in this framework is known as the evidence lower bound (ELBO), which is the final objective function that is used for the optimization of the active SLAM problem during training. This optimization function has two terms: the expected log-likelihood and the regularization term.

Dera *et al.* also employed the famous Taylor series for the activation function (non-linear transformation), which is known as an infinite series summation [7]. Therefore, the whole expression can't be used because an infinite number of access memory isn't possible. However, approximating the Taylor series to the first or second order is possible and still showed promising results. This paper scopes the Taylor series down to the first order. Thus, the first-order Taylor series approximation is up to the first derivative of the Taylor series formula, while the rest of the equation is a percent error. The mathematical equations were then derived from this expression (first-order Taylor series) in computing the next mean and covariance values for the next layer of neurons inside the network. The standard Taylor Series expansion 2.5:

$$f(z) = \sum_{n=0}^{\infty} \frac{f^n(a)}{n!}(z-a)^n \tag{2.5}$$

Furthermore, the main idea is that only the mean and covariance are propagated throughout the network. This creates each layer to have its own mean and covariance (distribution) through the network and, respectively, at the output layer, which will represent the prediction (mean) and the uncertainty of that prediction (covariance). Thus, if the covariance has a very small value, the model will be certain about the prediction. Otherwise, if the covariance has a large value, the model will be uncertain about the prediction. Therefore, the uncertainty is used as a measurement of whether the model will be certain or uncertain about a prediction. Having uncertainty is critical when measuring how certain the model is and, thus, is very beneficial when making crucial decisions.

The PremiUm-CNN model worked more toward a classification approach in which the output was to classify the probability of an object while also computing the uncertainty of that probable prediction [7]. The paper showed that the model was trained on several image datasets that provided promising results in how uncertainty improved the overall prediction in classifying images,

radar scans, medical, etc. This thesis project does not aim toward the classification approach but rather an action prediction approach in a DRL scenario in which the network's output is discrete actions instead of the classification of an input image.

CHAPTER III

METHODOLOGY

This thesis adopts the actor-critic (A2C) method instead of DQN for deep reinforcement learning on the agent/robot. This is mainly due to the flexibility of the A2C method for robots, as mentioned and discussed in the literature review section. A depth camera and a LiDAR will be equipped on the agent/robot. The camera's depth image/frame will be used as input to the DRL neural network to determine the best possible action based on the state of the environment. A LiDAR could also be used as input to the network, as seen in previous research, but a camera is more suitable for detecting objects and determining the best action along the robot's trajectory. In a sense, this is better for object detection or collision avoidance. The LiDAR has trouble detecting objects but is great for detecting the robot's surroundings/environment by a 360-degree laser measurement. The camera has trouble accurately measuring distance, but it is great for computer vision. Both have pros and cons, but ultimately, they work together to achieve the goal of the active SLAM problem in exploring unknown environments.

The following sections discuss the A2C structure, the idea behind implementing Bayesian inference to the A2C, Variational Inference (VI), KL divergence between the true and approximation distribution, the evidence lower bound (ELBO) objective function within the total loss function, the first-order Taylor series approximation, and the mathematical derivation used for the Bayesian A2C.

3.1 The Actor Critic (A2C) Structure

We use convolutional neural networks (CNNs) in both the actor and critic models to extract useful features or entities from an image/frame, which is a standard method for object detection or recognition using cameras and video footage. An image could be an RGB image, a depth image, or

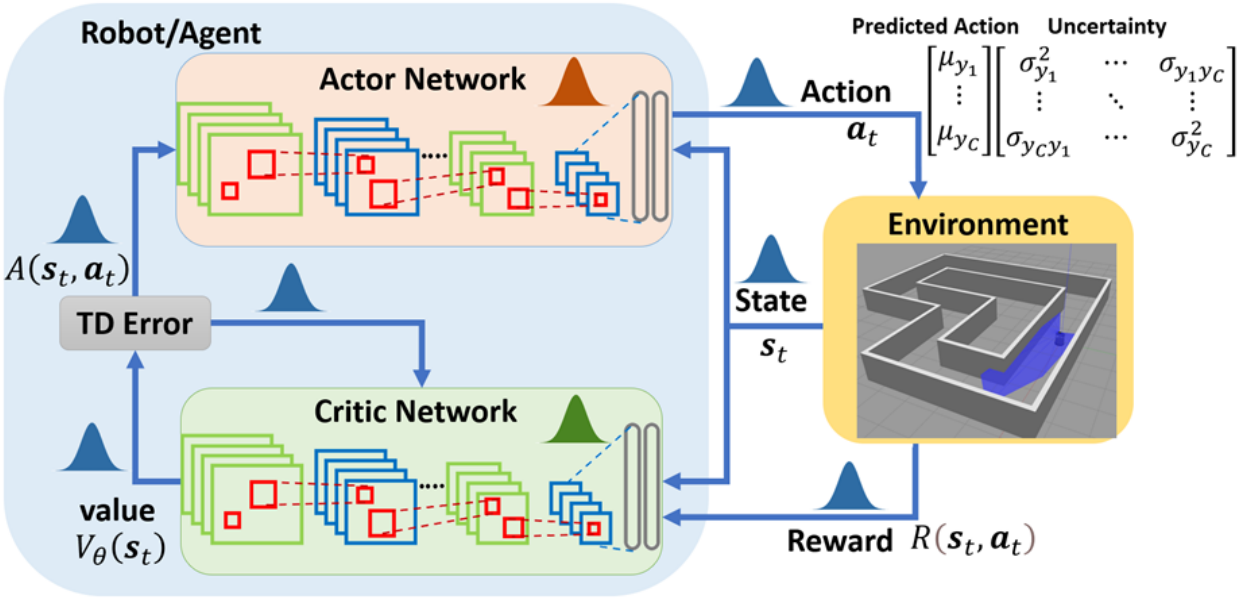

Figure 3.1: The proposed Bayesian A2C model framework consists of two convolutional neural networks: the Actor network and the Critic network. The parameters of both networks are treated as random variables that follow a variational posterior distribution. By propagating the mean and covariance of this variational distribution through the non-linear layers of both networks, we obtain the predictive distribution over the action written as $p(\mathbf{a}_t|\mathbf{s}_t, \mathbf{D})$. The mean and covariance of this distribution represent the predicted action of the robot and the associated measurement of uncertainty.

a pixel image. The two CNNs are connected through the temporal difference (TD) error function. We develop the Bayesian inference for the actor CNN to estimate uncertainty in the robot's actions. The uncertainty is measured by the covariance matrix of the predictive distribution. The mean of the predictive distribution represents the probable action to take. We adopt the PremiUm-CNN model [7] for our proposed Bayesian A2C model. Therefore, the proposed thesis combines CNN, A2C, and Bayesian inference with uncertainty. Figure 3.1 demonstrates the proposed Bayesian actor-critic (A2C) reinforcement learning model.

We set up the actor and critic convolution neural networks with a consistent learning rate and batch size of 16. The input image/frame size is set at 32×32, and the output layer of the actor network will classify predictions into three actions: moving straight, right, or left, using the Softmax function. The CNN architecture comprised 10 convolution layers, with the beginning three layers containing 32 kernels of size (5×5), the following three layers with 64 kernels of size (3×3), and

the subsequent three layers with 128 kernels of size (1×1). The final layer was fully connected with softmax function. The output of the actor network as shown in Figure 3.1 consists of the mean and covariance matrix. The mean indicates the predicted action, while the covariance matrix measures the uncertainty associated with the predicted actions. The actor's mean and covariance are passed to the environment which will give a corresponding reward. This process involves propagating the mean and covariance through the network using Bayesian inference, variational inference, KL divergence, ELBO objective function, advantage function, and an activation function to produce an action and uncertainty at the output.

3.2 Bayesian Formulation

The CNN used in this thesis is not a traditional deterministic CNN as seen where the unknown parameters, weights, and quantities are set to a single set of real values. Instead, a Bayesian approach is considered and implemented as a Bayesian CNN model where the unknown parameters and weights are set to random variables with a prior probability distribution. This means that all convolution kernels are random variables, resulting in all extracted feature maps, multiplication, activation functions, logits, and softmax also being random variables. The kernels in the CNN and the weights in the fully connected network are assumed to be independent of each other. Every layer propagates the mean and covariance matrix. The independent assumption within and across layers ensures that the feature maps can stand on their own without any repetition.

The Bayesian actor network incorporates prior knowledge about the parameters Ω before observing the data, denoted by $p(\Omega)$. The likelihood of the process probability generated by the data $\mathbf{D}$, given an Ω, is represented by $p(\mathbf{D}|\Omega)$. After observing the data $\mathbf{D}$, the posterior probability of the parameters Ω being true, which represents the total knowledge, is denoted by $p(\Omega|\mathbf{D})$. This posterior is calculated using the well-known Bayesian theorem and is referred to as the probability of a hypothesis given the evidence, $p(\mathscr{H}|\mathrm{E})$. However, in the context of machine learning, the focus is on the robot collecting data as evidence, and thus, the probability is expressed as the probability of Ω given the data, as shown in Equation 3.1. Although this posterior captures the total knowledge and is something desirable to calculate, it is intractable to calculate the posterior due to the large

dataset and the high-dimensional parameter space within the DNN, which has a vast of parameters and layers of nonlinear activation functions.

$$p(\Omega|\mathbf{D}) = \frac{p(\Omega)p(\mathbf{D}|\Omega)}{\sum_{\theta} p(\Omega)p(\mathbf{D}|\Omega)} \tag{3.1}$$

When dealing with large networks with many parameters, using Bayesian inference can be mathematically intractable. However, this issue can be solved by approximating the posterior. To address this problem, variational inference (VI) was adopted as a solution. VI uses a parametric distribution known as the variational distribution to approximate the true posterior distribution. This approach transforms high-dimensional density estimation into an optimization problem that minimizes the KL divergence between the approximation and true posterior distribution.

3.3 Variational Inference

Moreover, minimizing the KL divergence using the ELBO objective function helps reduce the distributional distance between the approximation and the true posterior distribution. The ELBO is derived from the KL divergence and variational inference and is used to obtain optimal parameters of the approximate posterior distribution by minimizing the difference between the approximation and the true posterior. Essentially, the ELBO approximates the true posterior with the variational posterior.

The ELBO in this context has two critical components: the expected log-likelihood from the training dataset and a regularization term. The mean and covariance of the variational distribution propagate through the network's layers. This applies not only to the CNN network but also to the two independent CNN networks that branch off, i.e., the actor and critic networks, leading to the creation of the proposed Bayesian A2C in this thesis.

The actor-critic model is a type of temporal difference (TD) learning approach that separates the policy function from the value function. The policy function is responsible for determining the agent's actions based on the current state, while the value function estimates the expected return for an agent starting at a specific state and following a particular policy. In the actor-critic method,

the policy is known as the "actor," suggesting a set of possible actions for a given state, while the estimated value function is called the "critic," which evaluates the actor's actions based on the policy.

To implement this method, we use CNNs for both the actor and critic models, and to incorporate prior knowledge, we introduce a prior distribution over the actor network's weights as $\Omega \sim p(\Omega)$. We applied variational inference to approximate the variational posterior distribution of the weights given the data, written as $p(\Omega|\mathbf{D}_t)$. Here, $\mathbf{D}_t = \{\mathbf{a}_t, \mathbf{s}_t\}$ represents the dataset, where $\mathbf{s}_t$ and $\mathbf{a}_t$ are the state and action at time t. This involves optimizing the ELBO objective loss function $\mathscr{L}(\phi; \mathbf{D}_t)$, as shown in Equation 3.2.

$$\mathscr{L}(\phi; \mathbf{D}_t) = E_{q_{\phi(\Omega)}}\{\log(p(\mathbf{D}_t|\Omega)\}) - KL[q_{\phi(\Omega)}||p(\Omega)] \tag{3.2}$$

Here, the variational parameters ϕ refer to the mean and covariance of the variational posterior $q_\phi(\Omega)$. T represents the the total training steps per epoch, ranging from 1 to T.

The ELBO's objective function $\mathscr{L}(\phi; \mathbf{D}_t)$ is composed of two components. The first part corresponds to the negative expected log-likelihood of the training data (environment maps) given the network parameters. The second part involves a regularization term, which is defined by calculating the KL divergence between the proposed variational distribution $q_\phi(\Omega)$ and the prior distribution $p(\Omega)$. During the training process of the Bayesian A2C networks, the ELBO loss function is minimized using the gradient descent update rule as part of the total overall loss function. The overall total objective function of the Bayesian A2C model can be expressed as Equation 3.3.

$$\mathscr{J}(\phi; \theta) = \sum_{t=0}^{T} \mathscr{L}(\phi; \mathbf{D}_t) A(\mathbf{s}_t, \mathbf{a}_t) \tag{3.3}$$

$$A(\mathbf{s}_t, \mathbf{a}_t) = \mathscr{R}(\mathbf{s}_t, \mathbf{a}_t) + V_\theta(\mathbf{s}_t + 1) - V_\theta(\mathbf{s}_t) \tag{3.4}$$

We define $A(\mathbf{s}_t, \mathbf{a}_t)$ (Equation 3.4) as the temporal difference (TD) loss. $\mathscr{R}$ are the rewards, $V_\theta(\mathbf{s}_t + 1)$ is the average value of the future state, $V_\theta(\mathbf{s}_t)$ is the value for the current state, and θ are the

parameters associated with the Critic network. The Critic network performs all value placements and advantage function calculations. This allows the Critic to critique the Actor's predicted actions in a state, letting it know which actions are bad and which are good or the advantage of a particular action over the others in a given state. In this way, the Critic network guides the Actor in making proper adjustments in order to take better actions next time around. Figure 3.1 provides a schematic representation of the Bayesian A2C model, showing both the Actor and Critic networks' roles in the full framework.

3.4 Mathematical Derivation of the Uncertainty Propagation Framework

We adopt the first-order Taylor series (TS) to propagate the mean and covariance matrix of the variational posterior through the non-linear activation function of every layer. The reason for choosing the first order was due to the fact that the TS can be an infinite summation of terms by the function's derivatives at a single point. Storing all of the values/variables would require an infinite amount of memory access, which realistically is not practically achievable. Additionally, processing all of the infinite data would be computationally heavy as well. Although this approach could have used higher orders such as second or third order or even higher orders, the higher the order of terms in the network, the more memory and process are required. This would make backpropagation take longer, as there would be more variables to adjust and tune, leading to more computational overhead for the TS summation. Instead, an estimation was deemed a better approach for memory efficiency and computational ease on the processor, and therefore, the first order was considered for this thesis.

In our model, we efficiently propagate the mean and covariance matrix, from the input to the output of the actor network after passing them through non-linear activation functions (the first-order TS) in each layer of the network. Consider a layer in the network, as shown in Figure 3.2. Here, $\mathbf{x}$ represents the feature maps before the activation function, and $\mathbf{z}$ represents the feature maps after the activation function f. f is defined as $\mathbf{z} = f(\mathbf{x})$. The activation function f used for each layer in the network represents the first-order TS approximation. This allowed the model to propagate the mean and covariance from $\mathbf{x}$ through f to obtain the next layer of feature maps $\mathbf{z}$. The mathematical formula used for both mean and covariance in between layers is represented as follows (Equation

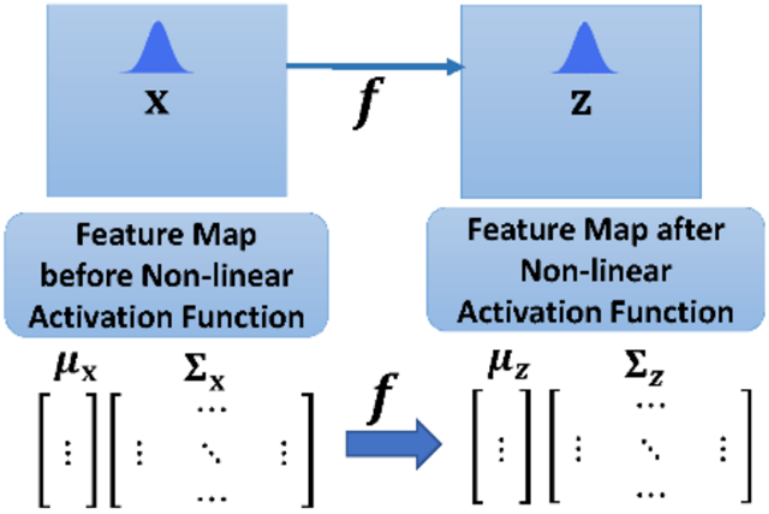

Figure 3.2: An example of a layer in the proposed Bayesian A2C network. Illustrates on how the mean and covariance matrix are propagated through each layer using the first-order Taylor Series approximation. Thus, the first-order Taylor series approximation for the non-linear activation function.

3.5):

$$\begin{aligned} \mu_{\mathbf{z}} &\approx f(\mu_{\mathbf{x}}) \\ \sum_{\mathbf{z}} &\approx \sum_{\mathbf{x}} \odot (\nabla f(\mu_{\mathbf{x}}) \nabla f(\mu_{\mathbf{x}})^T) \end{aligned} \tag{3.5}$$

The mean and covariance can be derived in any non-linear function layer within the model. By propagating the mean and covariance of the variational distribution, a predictive distribution for a new action $\hat{\mathbf{a}}$ given a state $\hat{\mathbf{s}}$ and the training data set $\mathbf{D}$ can be obtained. The distribution $p(\mathbf{a}_t|\mathbf{s}_t,\mathbf{D})$ is obtained by integrating out all possible values of the model parameters Ω. Therefore, the mean of the distribution, $p(\mathbf{a}_t|\mathbf{s}_t,\mathbf{D})$, represents the robot's intended action which could be a step or a movement, while for the covariance matrix captures the level or measurement of uncertainty associated with the predicted action. Thus, the closer the covariance value is to zero, the more certain the model is, but the further away the value is from zero, a measurement, the more uncertain it is about the action. Furthermore, both variables are produced at the final layer/output of the Bayesian A2C network.

The estimated uncertainty is a useful tool that can act as a warning signal, which allows the

system to be self-aware of noisy signals in the predicted action at the output of the actor network. Or better yet, it can send a flag or signal to other human users to make a proper adjustment outside the robot's scope. This reduces errors and improves precision, influencing the model to choose a better action depending on the certainty of the action and state. Thus, the agent can act and respond better in unknown or noisy environments.

3.5 Reward Setup

The reward function (Equation 3.6) consists of a standard non-collision method that harshly penalizes the robot in case of a collision or slight bad behavior. In all other cases, the reward system sends a large positive reward when the robot moves straight, a small reward if the agent turns, and a reward when it explores successfully around the environment, completing the map representation in both turning and forward actions. To achieve this, the optimal criteria function from TOED was used to reward the agent when it successfully explored and mapped unknown areas of the environment. Initially adopted from the active SLAM cost function J mentioned in the review of the literature (Equation 2.3), the exploration reward is bounded by the tanh(.) function to set a range for the SLAM cost function from 0 to 1, to avoid overshadowing the constant values in the turn or forward action rewards. Positive rewards are received by the agent when the robot moves forward without any collision and/or explores unseen areas, and penalization (negative rewards) when there is a collision. This teaches the agent to differentiate between good and bad actions based on the state it is in and learn from these experiences and data set (batch size of 16) collections during training. The reward function is shown as follows:

$$r\left(s_t, a_t\right) = \begin{cases} \text{-100} & \textit{if collision} \\ \text{1 + tanh(η/D-opt)} & \textit{if straight} \\ \text{-0.1 + tanh(η/D-opt)} & \textit{if turning} \end{cases} \tag{3.6}$$

During the training phase, the reward function was used. The D-optimally exploration reward was

derived from TOED. The robot was penalized whenever a collision occurred, and a positive reward was given when the robot moved forward and/or explored successfully without any collision.

CHAPTER IV

EXPERIMENTS AND RESULTS

In our experimentation, we utilized the Gazebo simulator and an open-source OpenAI gym extension called gym-gazebo. The gym-gazebo extension library has pre-built environments and robots that are readily available for use with machine learning architecture methods. However, we developed our own custom-proposed architecture/network known as Bayesian A2C and only used the pre-made Turtlebot3 robot and the pre-built environments. The software framework, libraries, messages, and tools for both the robot and its environment were constructed using the Linux operating system and the Robot Operating System (ROS).

To compare the performance and improvement of the proposed Bayesian A2C model against the deterministic (DET) A2C model, we trained, tested, and examined both models within a simulated Gazebo environment. During testing, we evaluated the robustness of the Bayesian A2C model by introducing random Gaussian and adversarial noise to the images/frames, simulating noisy environments that may be encountered in real-world applications. This noise analysis was performed for both the proposed Bayesian and DET models to ensure a fair comparison of up to 200 epochs/episodes.

For training the Actor and Critic convolutional neural networks (CNNs), we employed a consistent learning rate and batch size of 16. The input image/frame size was set to 32×32, and the output of the Actor network was classified into three actions: moving straight, right, or left, using the Softmax function. The CNN architecture consisted of 10 layers, with the first 3 layers having 32 kernels (5×5), the following 3 layers having 64 kernels (3×3), and the subsequent 3 layers having 128 kernels (1×1). The final layer was fully connected. To evaluate performance and robustness, we employed the moving average and cumulative rewards metrics throughout each run of epochs.

4.1 Experimental Setup

The environment will be set up in the Gazebo simulator. However, at the beginning of executing the network, OpenAI Gym and open-source code libraries are used to load up pre-defined environments to quickly develop the proposed algorithm (Bayesian network model) and avoid the tedious work of creating a robot and environments for the robot in the robotic operating system (ROS) framework.

The Turtlebot3 robot is used in this approach for simulations to train the network model and a deterministic model. After the simulation phase or after training, both models are deployed into a test environment and compared to each other in terms of their performances, robustness, convergence, and stability through the rewards received. The Turtlebot3 is a great robot for training and testing different methods for the active SLAM problem, and it can be easily deployed and tested from OpenAI library environments that are pre-made and pre-setup in the ROS framework. To import these pre-defined environments, robots, and ROS setups, the Linux operating system (OS) was installed.

Linux is a great OS to run machine learning models in Python and is flexible in allowing the user to install repositories, dependencies, third-party libraries, etc., for any robotic project. If equipped with an NVIDIA graphics processing unit (GPU), it can accelerate the machine learning process in deep learning. This is because NVIDIA GPUs already have CUDA cores pre-built and can easily install cudaDNN and libraries that are compatible with these GPUs. This benefit from the NVIDIA GPU allows us not only to have access to CUDA cores but also to easily have our machine with hardware acceleration turned on to speed up the processor in a heavy-duty task. This helps in running training procedures for our model in machine learning environments during both the training and testing phases.

The ideal environment setup is to create at least two simulated environments for training and testing. Both models, the deterministic A2C model and our Bayesian A2C model, were trained in the same environment with the same number of episodes to ensure a fair and identical comparison. The test phase for both models was also conducted in the same environment, with the same amount

of noise added to the environment. This was done to examine the performance and robustness improvements between the two models. The testing began with the weakest noise and gradually increased the noise complexity of the environments until the noisiest level was reached. Two different types of noise were tested: Gaussian noise and adversarial noise. Both types of noise were gradually increased in level to analyze the robustness of both models under unfamiliar, noisy conditions.

The proposed framework was deployed and tested on a simulated turtlebot3 robot. The experimental results and procedures were documented and analyzed using other methods. To analyze and examine the improved performance and robustness of the model when it undergoes noisy conditions or corrupted images, the rewards of both models were graphed. This was done to show the improvement of the Bayesian model over the traditional deterministic model and how uncertainty can improve the model's prediction in such noisy environments.

The thesis initially started from a simple starting point: an open-source code library, OpenAI extension (gym-gazebo). This was to ensure that all the software, repositories, and dependencies were installed correctly and to do at least basic machine learning tasks. The open-source code uses basic deterministic networks that are already pre-coded and ready to use. However, this is just a starting point to make sure that the core setup, such as the environment and robot, is up and running. From here, the proposed framework proceeded to be built, developed, and executed. This involved modifying the artificial network within the robot itself. Thus, we created our network/framework, the Bayesian A2C.

Furthermore, after training our proposed model, the Bayesian A2C, and the traditional deterministic A2C, we compared the two models during testing. We wanted to see how well the Bayesian model improves in performance and robustness, especially when the images are interrupted or corrupted by a noisy environment. This was done by purposefully adding Gaussian and Adversarial noise.

4.2 Training

The model's performance was tested and validated in multiple training iterations. The training aimed to teach the model how to navigate, explore, and build the map, also known as the active SLAM problem. Two models were utilized for the training: the deterministic A2C and the Bayesian A2C. Both models underwent 200 epochs of training, taking 1500 steps in each epoch. To ensure a fair comparison between them, they were trained using the same environment and the same robot. During training, the Bayesian A2C outperformed the deterministic A2C in gathering more rewards during each episode. However, the deterministic A2C learned its optimal action policy faster than the Bayesian A2C, resulting in a trade-off of longer training times but better performance and stronger robust actions. Moreover, it took less time to train the deterministic A2C than the Bayesian A2C because the Bayesian A2C consists of propagating the variational distribution throughout the framework, which results in more weight parameters to tune and reinitialize in the model.

Figure 4.1 illustrates the training of both models: the Deterministic (red) and Bayesian (blue) A2C. The y-axis represents the average rewards, and the x-axis represents the number of steps throughout the run. The graphs demonstrate that the deterministic A2C reaches its optimal policy faster than the Bayesian A2C. Therefore, the research measured the time it took to train both models. Three different training runs were done for both models, and the average training time for the deterministic A2C was about 2.99 hours, while the average training time for the Bayesian A2C was about 4.08 hours.

The longer training time for the Bayesian A2C is due to the fact that there are more network parameters to reinitialize and tune. However, this drawback is offset by the fact that the Bayesian model performs better and has far better robustness in predicting actions to maneuver, navigate, and explore the environment compared to the deterministic model. The results were examined and compared for both models during the testing phase, where noise was purposely added, as discussed in the next section.

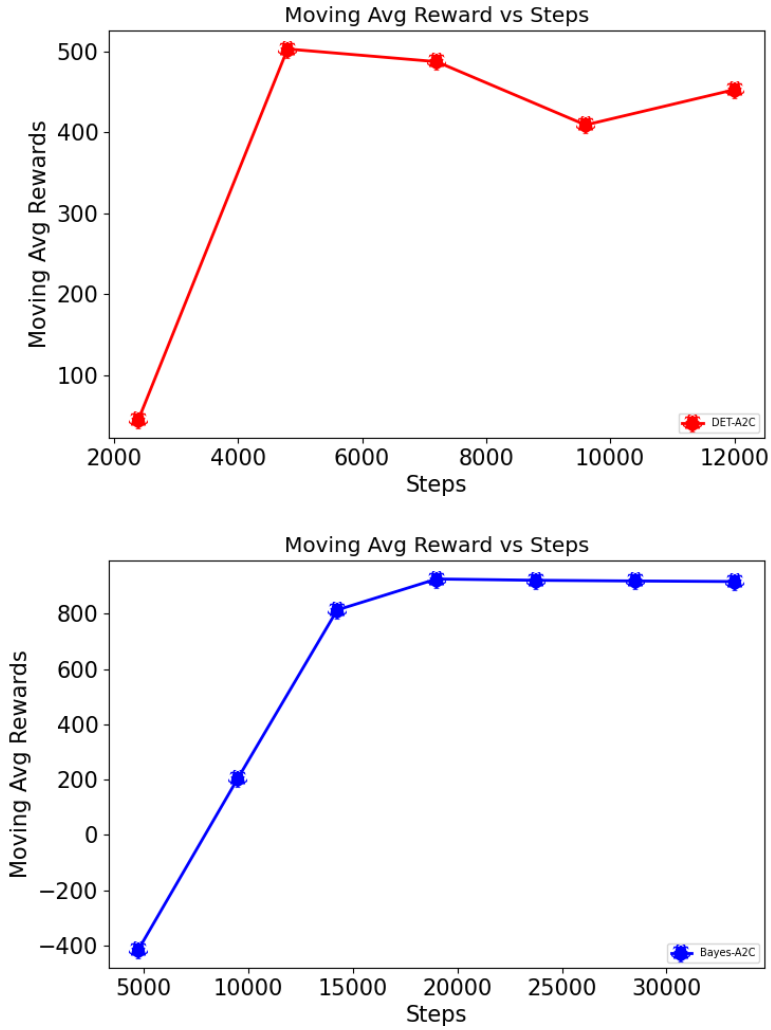

Figure 4.1: Training results for deterministic A2C (red) and Bayesian A2C (blue) in terms of moving average reward.

4.3 Robustness & Noise Analysis

4.3.1 Gaussian

During testing, Gaussian noise was applied, and the results were examined at seven different levels of noise. Only the standard deviation (σ) was modified from the Gaussian formula, with the mean (μ) kept at zero throughout all levels. Below is the well-known function shown in Equation 4.1:

$$f(\mathrm{x}) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-(\mathrm{x}-\mu)^2/(2\sigma^2)} \tag{4.1}$$

By adjusting only the standard deviation (σ), we were able to apply multiple different levels of noise onto the images for both models. This adjustment allowed us to apply a little to a large amount of noise or blur to the input images, where the standard deviation value ranged from 0.0001 up to 0.5. Furthermore, we tested seven different noise levels [0.0001, 0.001, 0.1, 0.2, 0.3, 0.4, 0.5], each with a total of 200 epochs/episodes per run. A moving average and cumulative rewards were also plotted in each run with a different noise level to examine the performance and, importantly, the robustness between the two models: Bayesian A2C and the deterministic A2C. In total, we tested 7 different noise levels.

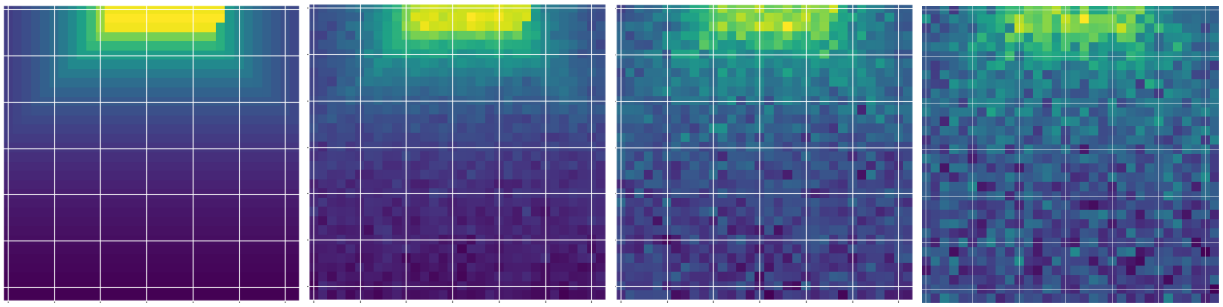

Figure 4.2: Examples of clean and noisy images used during testing. Starts left with a clean image, then 0.1 noise level, then 0.3 noise level, and lastly 0.5 noise level.

In Figure 4.2, the noise levels are shown. The images start from the left with no noise (clean) and gradually increase towards the right. The first image on the left is clean, while the last image on the far right has the highest noise level of 0.5. As observed, the images were intentionally made noisy and blurry to assess the model's performance in each instance. We tested all noise levels on both models and analyzed the enhanced robustness and performance of the proposed model.

First, this thesis discusses and examines the test performance for the standard deterministic (DET) A2C in Gaussian noise (top graph), Figure 4.3. The line plot red represents noise level 0.0001, blue represents 0.001, green represents 0.1, yellow represents 0.2, black represents 0.3, cyan represents 0.4, and purple represents 0.5 for both graphs respectively. The deterministic A2C did

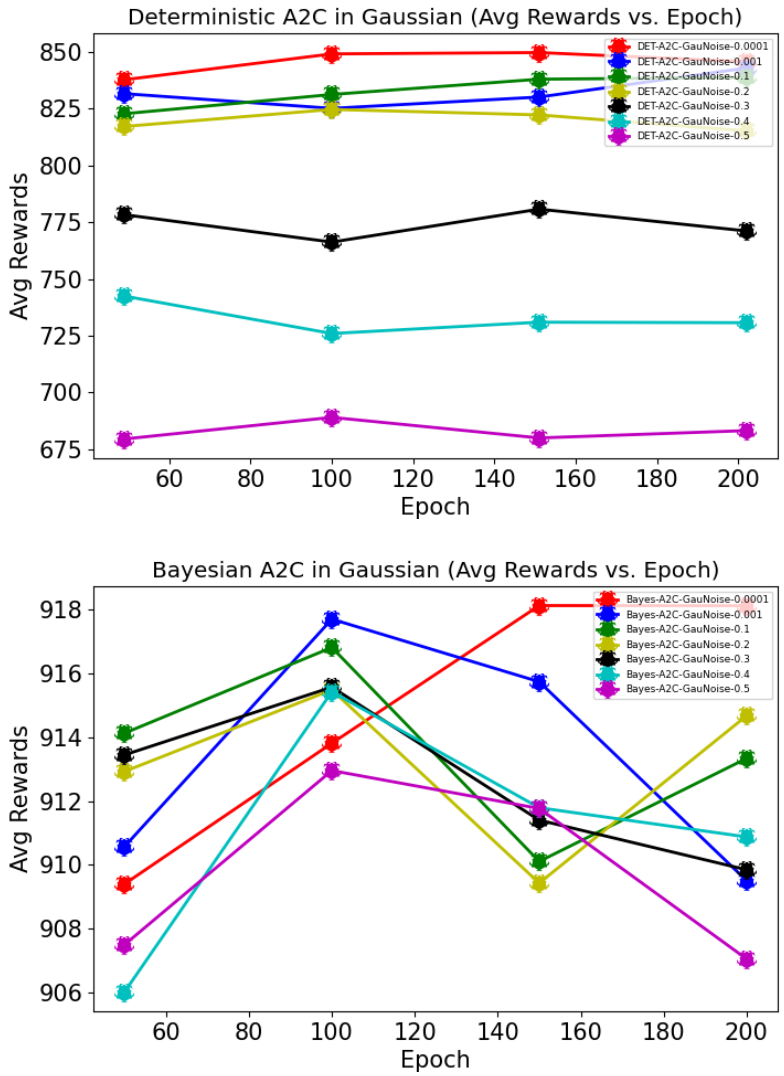

Figure 4.3: The experiment test results for deterministic A2C (top) and Bayesian A2C (bottom) in all Gaussian noises.

well in mapping, navigating, and exploring when placed in the test environment. The agent received acceptable test results in rewards throughout the epochs, but this was only the case when the noise was very low and insignificant to what the model was trained on, such as the 0.0001 and 0.001 values. Only small drops in performance/rewards from these noises were seen. However, when the noise increased to a significant difference from what the model was trained and seen before, the performance dropped pretty significantly as well. The performance hit came around the 0.2 value but dramatically dropped when the noise value came to 0.3 and only got worse as the noise

increased. The gap between 0.3 and 0.4 dropped heavily in performance as compared to lower noise values. Then, the gap widened and dropped even more when the noise increased to 0.5, the highest noise value tested for this experiment. From the experiment results, we concluded the robustness of the deterministic was very poor whenever the noise level on the images changed. Depending on how noisy the interruption was, it affected the agent's performance badly. This can be appalling in cases in the real world where a robot may experience interruption, interference, or corruption but still wants to choose the best possible action or at least close to the best performance when there is no noise at all. Figure 4.3 shows the performance of the proposed Bayesian and deterministic models for all noise levels and shows the poor robustness of the deterministic model. The graph also plots the moving average reward through the 200 epochs tested through the run.

Next, this thesis analyzed the performance and robustness of the Bayesian (Bayes) A2C (bottom graph) in Figure 4.3 that used the same noise levels, test environment, and the same number of epochs in each run. Everything was done the same as done previously with the deterministic A2C to keep a fair comparison between the two models. The only thing different was, of course, the network model itself on the agent. The Bayesian A2C performed outstandingly not only in its performance but also in robustness, which was the main purpose and goal of this study.

At the lowest noise levels, the performance was remarkable, with a high gain of rewards, and it had little effect on the performance whether the noise was set at 0.0001 or 0.1. The model's reward started high but then dropped notably at the 0.2 noise level, as expected, and the intention was to examine how much the performance dropped. We assumed that the drop would only be minor coming into the experiment, and the results proved to be true.

At the 0.3 and 0.4 noise levels, the performance had a very minor decrease and stayed in a small range of reward values compared to the best case. The worst performance, however, was, of course, when the noise level was at 0.5, the highest value. Though even at the highest level, the agent still performed well with only about an 8-point reward drop. It's not much compared to the best performance. The bottom graph in Figure 4.3 showed as the noise increased, so did the drops, but very minor drops and stayed close to the best-case performance. Thus, the hit on the model

was insignificant and stayed close to the case where there was no noise, which was proposed in our hypothesis.

From the experiment, we analyzed the results and concluded the uncertainty measurement on the network helped the agent to choose better actions and predict better outcomes based on what it was certain or uncertain about. This can be especially true when noise levels increase greatly with only a slight performance drop due to the strong robustness of our proposed Bayesian A2C model. All the test noise results can be seen in Figure 4.3; the moving average reward was plotted for 200 epochs from all noise levels.

Next, the comparison between the Bayesian A2C and the deterministic A2C moving average reward was discussed and examined. The top graph and bottom graph in Figure 4.3 were plotted on the same graph to show the different performances between the two. The Bayesian A2C model was found to have performed better than the deterministic A2C, with an approximately 80-point better average performance. Additionally, the Bayesian model demonstrated better robustness throughout the noise levels, maintaining a high reward value above 900 at all levels with only a minor hit in performance. In contrast, the deterministic model was harshly affected by noise increase, resulting in significant drops in performance from one noise level to the next. Figure 4.4 illustrates the improved robustness of the Bayesian A2C and how adding an uncertainty measurement and self-awareness mechanism to the agent itself can really help the agent predict better actions when noise changes occur. Overall, the proposed Bayesian model demonstrated superior robustness and kept it to somewhat the same performance throughout all noise levels when compared to the deterministic model.

In Figure 4.4, the moving average rewards, performance, and robustness of the proposed Bayesian A2C model (solid lines) and deterministic (DET) A2C model (dotted lines) are demonstrated. The experiment shows the improved robustness achieved when noise was added to the images. Additionally, the raw cumulative rewards were also tested and plotted, but for clarity, only the moving average results are presented here. It's worth noting that the cumulative rewards exhibited a similar relationship and behavior with the moving average, which led to a focus on the

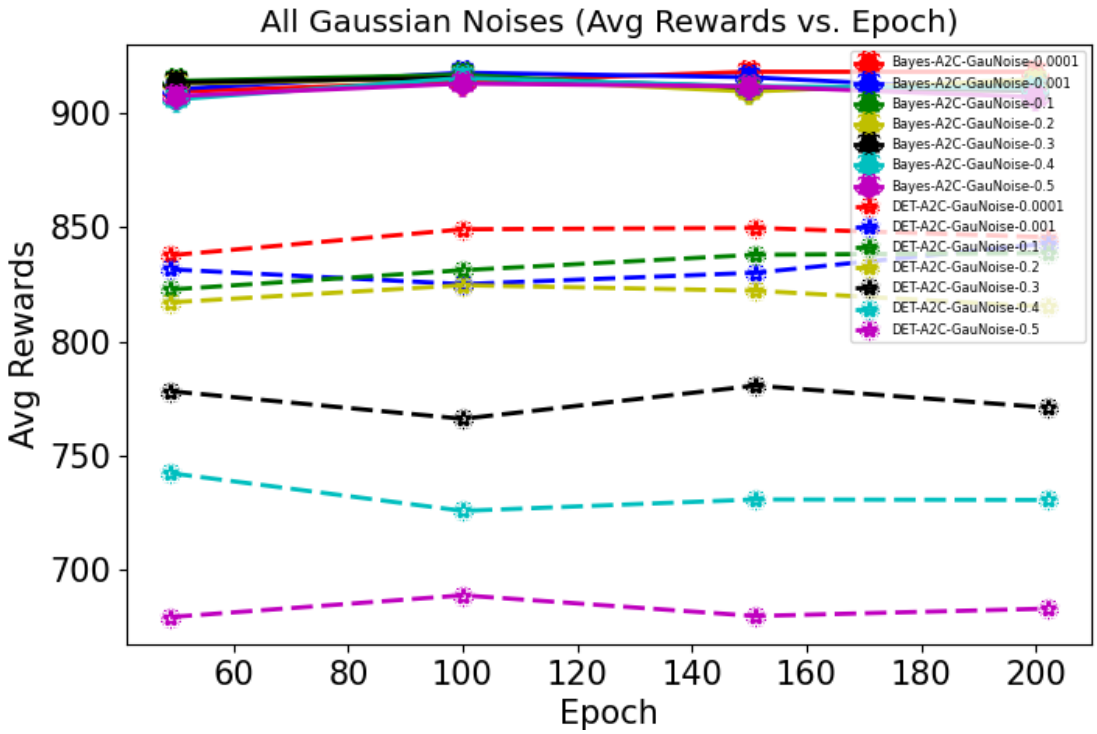

Figure 4.4: The experiment test results for both deterministic A2C (dotted lines) and Bayesian A2C (solid lines) in all Gaussian noises.

average for analysis.

Notably, the moving average and cumulative rewards of the DET model were significantly higher in the clean environment compared to the noisy environment, with rewards sharply decreasing as the environment became noisy. This was not the case for the Bayes model, which stayed at a very high reward rate for all noise levels and had strong robustness. The Bayes model's performance only changed slightly as the noise level in the environment changed greatly, while the same was done for the DET model. Therefore, the Bayes model proves to be more stable and robust.

4.3.2 Adversarial Noise/Attack

Additionally, during testing, the input images were subject to adversarial attacks. Adversarial noise differs from Gaussian noise in that it teaches a model to employ various tactics to deceive and mislead our network (the Bayesian A2C and deterministic A2C). Essentially, it acts like an attacker towards the model itself, attempting to mislead the agent into generating improper actions.

The primary concept here is that the adversarial model attacker purposely attempts to make our model predict incorrect actions or cause unexpected behavior and learn from those interactions to further deceive our model even more with each iteration. As a result, minor or significant changes in the input images can quickly deceive the model into predicting erroneous actions by determining which part of the image to blur, distort, or make fuzzy. This process enabled the research to analyze the strength of the robustness of these two models when adversarial attacks learn various ways to deceive them. Essentially, it is a technique to assess the flexibility of these models and comprehend their weaknesses and vulnerabilities when under attack and how well they can defend against them, hence, their robustness.

In the adversarial attack, changes in the noise level are different compared to the Gaussian noise. It involves adjusting the epsilon value. This variable was adjusted to modify the strength of the attack added to the images for each run and was used in the formula (Equation 4.2) to deceive our models by increasing the amount of additional noise, distortion, or blur that would be added to the images or parts of the images. The values of this variable ranged from 0.0001, representing the weakest noise, to 0.2, the strongest noise. Specifically, 6 different noise attacks and 6 different runs were tested, with values of [0.0001, 0.001, 0.01, 0.05, 0.1, 0.2] being employed. However, to maintain a valid comparison, 200 epochs were considered for each run.

$$\mathrm{X}^{FGSM} = \mathrm{X} + \varepsilon sign[\nabla_x \mathscr{J}(\mathrm{X}, \mathrm{y}_{action})] \tag{4.2}$$

Equation 4.2 represents the adversarial attack function used to generate the attacks on the model. The function uses a gradient descent method to learn effective ways to fool and trick the model by calculating the gradient of the model's output prediction (y_{action}) with respect to the input (X). Then, the gradient is multiplied by a constant value known as epsilon (ε), which is the variable that we modify in each run to change the level of attack. After multiplication, it is added to the original image to create a corrupted image that is fed to the model as input, resulting in the attack. Basically, in this approach, the attacker corrupts or modifies the original input through gradient descent to

deceive the model into making mistakes and predicting bad actions. The attack then learns to better fool the model since it uses the model's output and input to calculate the gradient to distort the original input.

In addition, this section explores the comparison of performance, robustness, and the relationship between the two models in the context of adversarial attacks, ultimately demonstrating the superiority of the Bayesian A2C over the deterministic A2C. This comparison is particularly noteworthy as it highlights the effectiveness of the proposed approach and underscores the superiority of the proposed model when compared to the deterministic standard model.

First, the deterministic A2C was examined in the context of adversarial attacks (top graph of Figure 4.5). The line plot red represents attack level 0.0001, blue represents 0.001, green represents 0.01, yellow represents 0.05, cyan represents 0.1, and purple represents 0.2 for both graphs, respectively. Based on the findings in the graph, the level with the lowest amount of noise demonstrated the best performance, reaching above the 880 reward mark. As expected, the level with the most amount of noise performed the worst, collecting about 840 rewards in each epoch. The same trend was observed with every noise level increase, causing a significant performance drop. Specifically, from the least amount of noise to the most amount of noise, there was about a 55-point reward drop. All noise levels are shown in the top graph of Figure 4.5, highlighting the robustness of the proposed model.

Next, the rewards collected in each epoch for the proposed Bayesian A2C (bottom graph of Figure 4.5) were also discussed. Here, the model performed well and maintained a high reward count at around 910 until the noise level increased to a very high value. The Bayesian model performed remarkably in almost all noise levels and had strong robustness in each level with not much change to its performance. From 0.0001 to 0.05, it remains almost the same, with a very minor performance hit. In this range, there were not many drops. However, when the noise level reached very high noise values of 0.1 and 0.2, a notable drop was seen. However, in the graph, there was a heavy performance drop that did not come until reaching the strongest noise tested. So, the experimental results still showed how well the robustness was in the range of the lower

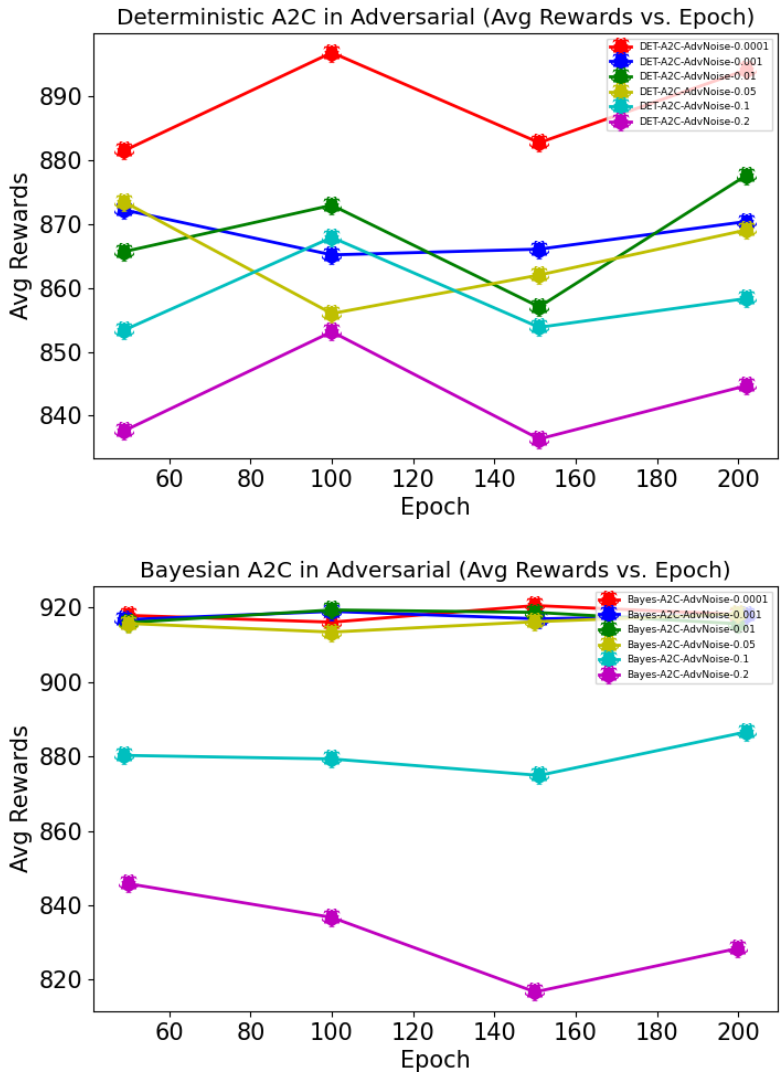

Figure 4.5: The experiment test results for both deterministic A2C (top) and Bayesian A2C (bottom) in all adversarial attacks.

and mid-noise levels. Even in cases where noise was added to the model, it still performed well in collecting good reward values. The model performed well for most noise levels with reward values above 900, or even close to it in the case of 0.1 noise level. The average reward values for all the noise levels are shown in the bottom graph of Figure 4.5, demonstrating the strong robustness of the Bayesian model.

Furthermore, Figure 4.6 showcases the moving average reward, performance, and robustness of the proposed Bayesian A2C model (solid lines) compared to the deterministic (DET) A2C model

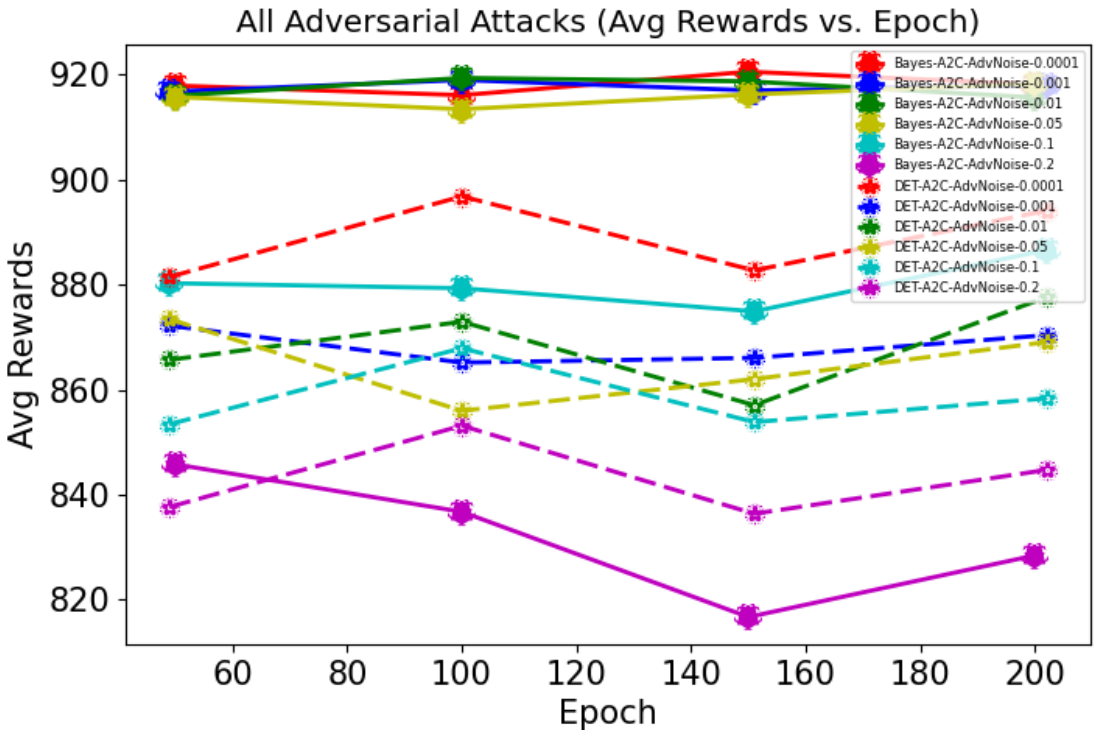

Figure 4.6: The experiment test results for both deterministic A2C (dotted lines) and Bayesian A2C (solid lines) in all adversarial attacks.

(dotted lines). The Bayesian A2C shows superior robustness, especially at mid-level noises. The performance drops only about 3 reward points during those mid-noise levels. Although there was a notable and drastic drop at the 0.1 level, the model still received a respectable amount of rewards and outperformed the deterministic model at that level. Despite the drop, the rewards were acceptable, and the performance still improved over the standard deterministic model. Another drop occurred at the 0.2 level from the Bayesian A2C and was on par with the deterministic A2C for the first time in the experiment test. However, in all other level cases, the Bayesian outperformed dramatically in achieving higher rewards and having little effect from these changes in noise levels.

Again, the same relationship happened here with what had happened in the Gaussian section, which showed enhanced robustness in most noise level changes. The raw cumulative rewards were also plotted, but for simplicity, the moving average was the only graph shown in this section. However, it showed the same aspects as the moving average, plus the moving average had a clear

illustration with a smooth line trend instead of a fluctuating or oscillating line from the cumulative rewards. The DET model showed instability again by dropping dramatically with every noise increase in the environment, while the Bayes model showed strong stability (or robustness) through most noise level changes. Bayesian A2C outperformed and improved over the DET model by achieving a very high reward rate and kept that performance through almost all noise levels with little to no change at all. This was tremendous to prove our claim and hypothesis in enhancing the robustness of models by including the uncertainty measurement on the agent itself. Thus, again, it showed how strong and stable the robustness was on the Bayesian A2C and backed up our assumptions and theory.

In concluding this section, we can see that the proposed model exhibited similar behavior when faced with either Gaussian or adversarial noise. However, the Bayesian A2C outperformed the deterministic A2C in almost every case and demonstrated improved robustness. The Bayesian A2C showed superior performance and robustness, with only minor reward drops in all cases. Adversarial noise proved to be stronger due to its ability to learn different ways of tricking the model and dynamically changing the noise being generated for the images. On the other hand, the Gaussian noise was treated like a constant value term or a static value being added to the images. Theoretically, this approach worked in our theory, and the experiments confirmed that it applied practically as well. These experiments were a success and showed that including uncertainty can improve the model's prediction and generate better actions based on what is certain or uncertain depending on the state it's in, whether it's noisy or not. The Bayesian model's performance was excellent regardless of whether the noise was constant or dynamically changing like an attack, with high reward values and only minor drop hits. Overall, this proved to be a successful experiment in what was proposed that demonstrated strong, robust behavior in all cases in the tested environment.

4.3.3 Uncertainty Analysis

Variance vs Signal to Noise Ratio (SNR). This portion discusses the relationship between the action variance and the signal-to-noise ratio (SNR). Since only the Bayesian A2C generates the variance and uncertainty measurement, this section examined only the Bayesian model due to the

fact that the deterministic model does not capture these quantities, leaving only the proposed model to analyze. The variance represents the model's uncertainty over the predicted actions. It measures how far or broad the prediction was compared to the mean of all past predictions. Each prediction value varies and is measured individually by how far apart each value is from the mean value point. Therefore, just like the spread of all vast value points quantifies the noise in the incoming signal. When the variance is low, it means that the individual value is very close to the mean from all values, which is wanted and favorable to this outcome. However, when the variance is high, it is quite the opposite. The individual value is further away from the mean and has a larger spread, which is not wanted because it signifies a larger noise on the image signal that may impact the final action output.

The SNR serves as a ratio of noise level added to the original image, or in other words, the ratio between the original image value and the noise value that is generated by the adversarial model/attacker. In essence, it is a ratio measurement of the quality of the signal relative to the noise by examining the image signal's strength to the noise strength generated. The formula uses a logarithmic expression on both the input signal (the image) and the incoming noise added, expressed as decibels (dB) terms. A low value of SNR represents a weak image signal compared to the stronger noise strength being generated. When the SNR value is high, the image signal is stronger than the incoming noise, implying a clearer image signal. Obviously, a higher SNR value is more desirable than a low SNR value because the model/agent can extract more useful entities or information from these images, and a clear image is better to use and see than a disturbed, noisy image. The experiment examined the relationship between different cases of SNR values (noisy images) with variance values to analyze the uncertainty predictions on actions and signals under a set of noise stress.

In Figure 4.7, the x-axis represents the SNR values evaluated during testing for all noise levels, while the y-axis represents the action variance also evaluated during testing for all noise levels. Since two types of noise, Gaussian and adversarial noise were tested, two line graphs were plotted to represent them. The dotted blue curve graph represents Gaussian noise and has 7 SNR

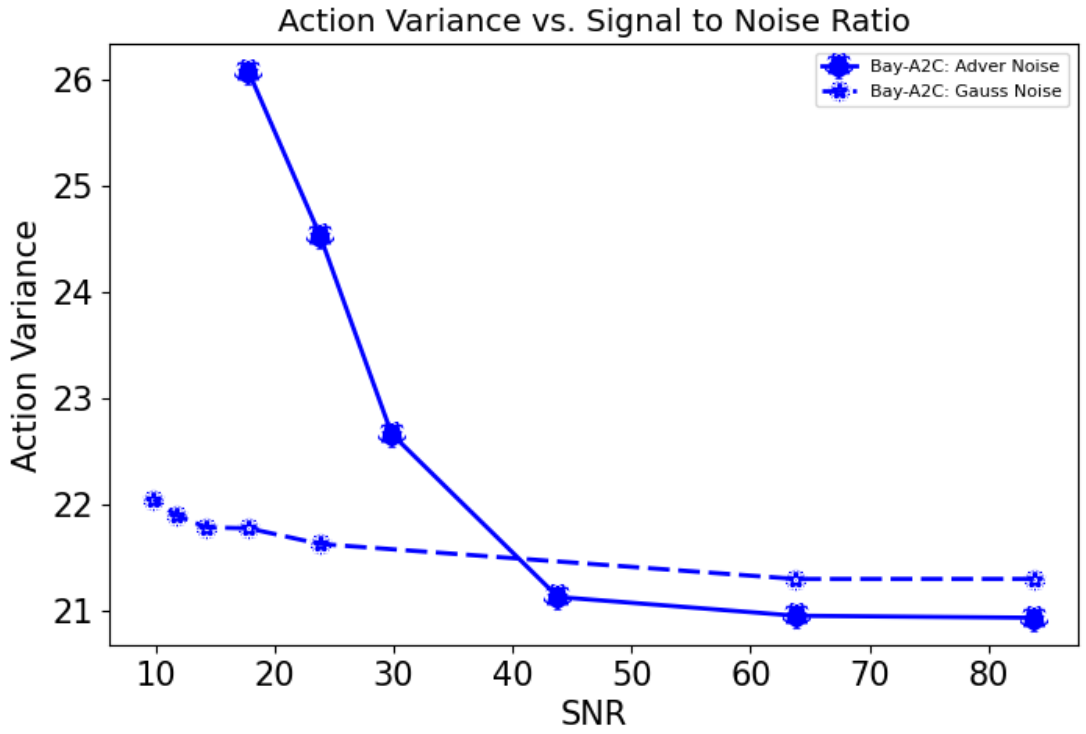

Figure 4.7: Action variance vs signal to noise ratio (SNR) graph. The dotted blue curve represents the Bayesian A2C under Gaussian noise, while the solid blue curve represents the adversarial attacks.

and variance data points plotted, namely [0.0001, 0.001, 0.1, 0.2, 0.3, 0.4, 0.5]. The solid blue curve graph represents adversarial noise and has 6 SNR and variance data points plotted, namely [0.0001, 0.001, 0.01, 0.5, 0.1, 0.2].

The model exhibited a gradual increase in the variance from the beginning to the end, especially for adversarial noise. Although the increase was not significant for Gaussian noise, it was enough to demonstrate the correlation between action variance and SNR as a representative of the noise level. The graph reads from right to left, where the most right SNR corresponds to a 0.0001 level of noise while the most left SNR value corresponds to a high noise level of 0.5. This indicated that initially, the SNR was high, and the variance was low. As the noise increased (equivalently, SNR decreased), the action variance value also increased. As previously mentioned, this was an anticipated outcome due to the relationship between these concepts related to noise. Nonetheless, this relationship was noticeable, especially in the adversarial plot.

The adversarial display showed an increase from the start (from right to left) of around 42 dB, almost like an exponential increase. This occurred because the graph began (from the right) with the lowest noise value (0.0001) and ended (left) with the highest noise value (0.2). As we mentioned earlier, the action variance was expected to be large when the image was noisy and small when the image was less noisy or in a clean state due to the predicted action and uncertainty. However, the opposite was true for the SNR, which was large when the image was less noisy but small when the image was extremely noisy. Hence, the image signal became strong as the noise became weak or small, as demonstrated on the line graph. Above the SNR value of 42 dB, the line became steady, indicating that there was less and less noise in the image. Then, the model became certain about its predicted actions as the noise level dropped. Finally, the action variance began to decrease while the SNR values began to increase.

It was expected that when images are noisy, the uncertainty in the actions would be higher. This means that the uncertainty would increase, or the action variance of the prediction would be larger, while the SNR would be smaller under noisy conditions. On the other hand, under clean conditions, the uncertainty would decrease. This was demonstrated on both line graphs, which experienced a decrease as the noise became less. The relationship between the variance and SNR is inversely proportional, while the variance is proportional to noise.

Reward vs Signal to Noise Ratio (SNR). In this section, the relation between the maximum and average rewards with respect to the SNR values for both Gaussian and adversarial noise is discussed. The maximum reward values were determined by the highest reward received throughout the 200 epochs in each noise level. The average reward values were calculated as the average of all rewards received from the 200 epochs in each noise level.

The max and average rewards were both plotted in a separate graph against SNR (Figure 4.8) for both Gaussian (blue line) and adversarial (red line) noise. The Gaussian and adversarial noise and the Bayesian and deterministic models were both plotted on the same graph for the analysis. To distinguish both models from each other, the Bayesian A2C was plotted as a solid line, while the deterministic A2C was plotted as a dotted line. As mentioned in the section before, the

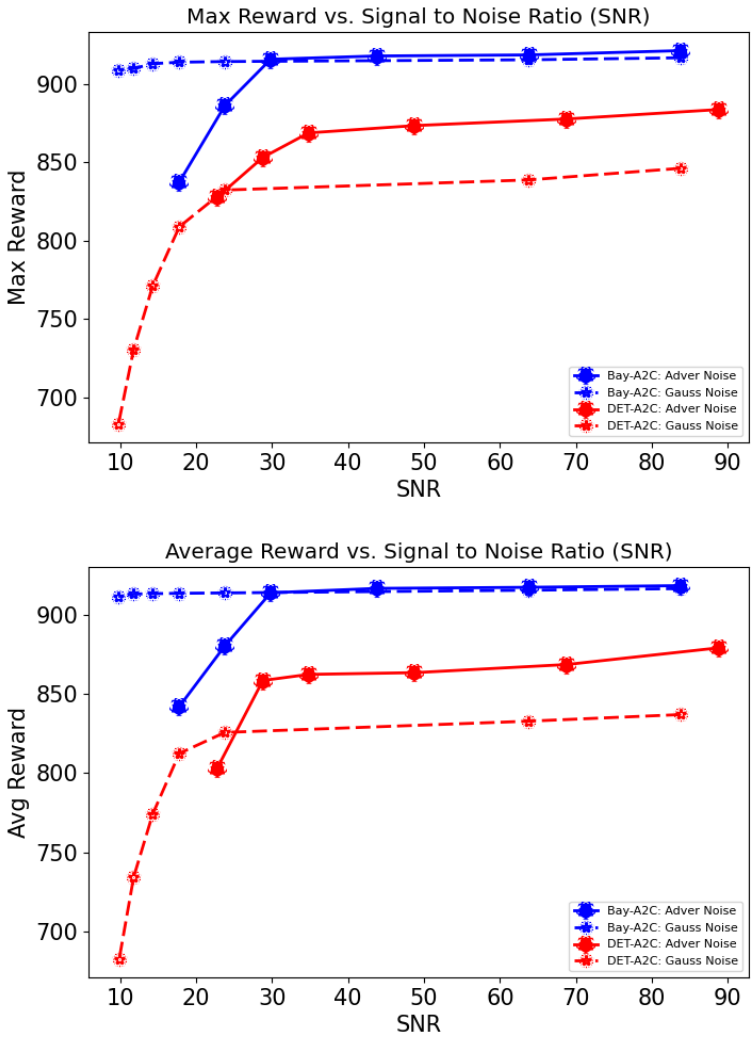

Figure 4.8: Maximum (top) and average (bottom) rewards vs signal to noise ratio (SNR) graphs. The blue curve represents Bayesian A2C under Gaussian noise (dotted line) and adversarial attacks (solid line), while the red curve represents deterministic A2C under the same noise conditions.

adversarial noise has 6 data points, and the Gaussian noise has 7 data points plotted since those were the amounts of noise tested.

It is noted that both the maximum and average values showed a similar resemblance. However, the only noticeable difference was at the left end of the graph, which was the low values of SNR (noisy) from the deterministic model on the adversarial noise (red dotted line trend). The average started lower around 800 and began to have a steeper change than the maximum values as the SNR increased. Whereas the maximum values started higher around 825 but began to have

more of a smooth exponential increase of change. The same applies to the Bayesian A2C model under adversarial noise (red solid line trend). Though the main difference between the max and average rewards is the models themselves, both values remain similar.

As previously mentioned, lower values of SNR indicate that the noise is stronger than the original input image, resulting in more noise. Higher values of SNR indicate that the noise is weaker than the image itself, resulting in a cleaner image with less noise. In both models, whether under Gaussian or adversarial noise, the rewards received decreased when SNR values were low, but the rewards received increased as the SNR increased. More noise led to fewer rewards, resulting in poor performance. However, less noise resulted in more rewards and better performance. In all cases where SNR was low, the models performed worse, and when SNR was high, the performance was at its best. This was expected, given that noise on the images affects the performance of the models, but to what extent and by how much?

In the case of the Bayesian A2C under Gaussian noise, the rewards received per 200 epochs were not significantly affected by the SNR values being either low or high. While the performance did drop from the highest to the lowest, it was only by about 5-10 points. In comparison, the deterministic A2C saw a significant drop of around 95 points from the highest reward to the lowest reward. This demonstrates the improved robustness of the proposed Bayesian A2C model, as it is able to maintain close to the best performance even under noisy conditions or when the SNR values drop. It stayed close to the best performance while the hit/drop was minimal, showcasing strong resistance to noise. Moreover, the performance of the model improved, reaching the 900 rewards mark, showing that it has both better performance and strong robustness to noise overall.

Under adversarial noise, the performance of the Bayesian A2C had a greater impact than that of the Gaussian test, especially at the lowest SNR values when the noise level was set at 0.1 and 0.2. This was unexpected, as adversarial attacks can learn to trick the model and force it to generate bad behavior or actions rather than compare Gaussian noise. Despite the high noise levels and low SNR values, the Bayesian model still outperformed the deterministic model and conveyed a more robust model throughout all the SNR points. Additionally, the deterministic A2C performance gradually

decreased with each SNR drop and dropped significantly at the SNR value of 30 (noise level at 0.1), which was a very high noise level for adversarial noise but worth testing. Although both models experienced a performance drop, the Bayesian's performance remained more consistent and steady on the mid/highest SNR values. Similar to the Gaussian test, the adversarial test also demonstrated the superior performance and robustness of the Bayesian model against different types of noise strength.

Table 4.1: All the Bayesian A2C Experimental Values for Gaussian Noise.

	0.0001	0.001	0.1	0.2	0.3	0.4	0.5
Variance	21.306	21.306	21.635	21.782	21.791	21.9	22.054
SNR	83.8	63.8	23.8	17.8	14.3	11.7	9.8
(Bay)Max Reward	916.7	915.423	914.288	913.864	912.926	909.846	909.011
(Bay)Avg Reward	916.316	915.313	913.596	913.379	913.127	912.864	911.365
(DET)Max Reward	846.197	838.76	832.301	808.853	771.164	730.738	683.2
(DET)Avg Reward	836.894	832.734	825.627	812.423	774.136	734.537	682.93

Table 4.2: All the Bayesian A2C Experimental Values for Adversarial Attacks.

	0.0001	0.001	0.01	0.05	0.1	0.2
Variance	20.941	20.96	21.136	22.67	24.535	**26.081**
SNR	83.811	63.811	43.81	29.831	23.81	17.79
(Bay)Max Reward	921.269	918.518	917.844	915.77	886.264	837.315
(Bay)Avg Reward	918.213	917.156	916.469	913.695	880.293	841.899
(DET)Max Reward	883.674	877.642	873.43	868.769	853.136	827.648
(DET)Avg Reward	878.989	868.454	863.324	862.221	858.373	802.962

Tables 4.1 and 4.2 provide important information on the data values collected during the experiments for both Bayesian and deterministic A2C in Gaussian noise and adversarial attacks. Table 4.1 presents all the different Gaussian noise levels tested, along with their variance, SNR, and reward values. This table allows for a direct comparison between the Bayesian and deterministic A2C values. These values were used for graphing purposes, but the table shows the actual raw values used for those graphs.

As the noise increases in each test run, the variance increases gradually while the SNR decreases, affecting both factors. Variance measures the model's uncertainty about the actions, and as variance (uncertainty) increases, the SNR decreases, becoming very noisy in each test run.

Additionally, the table shows the rewards and the relationship between each factor, with the rewards decreasing as noise increases, indicating how increasing noise affects performance. Therefore, increasing noise affects not only the uncertainty but also the performance of both models.

However, because the proposed model (Bayesian) includes uncertainty and a stronger robust feature on the model, the effect on performance is not significant compared to the deterministic A2C. This is also represented by the graphs in the previous section, Figure 4.7 and Figure 4.8, which illustrate the relationship between the variance and SNR and the rewards received and the SNR, respectively.

More importantly, the focus is mainly on Table 4.2, where adversarial attacks were introduced into the models. Similar to the Gaussian noise, the adversarial attack consists of reward drops, decreases in SNR values, and an increase in variance, all while the adversarial attack gradually increases. However, since adversarial attacks are known to be a harsher method of adding noise than Gaussian, due to the way it learns proper tactics to purposely trick and fool the model, the rewards drop even more, but the variance increases more as well. At noise attack 0.0001, the variance starts at 20.94, which is common at that noise range, but when the noise attack increases and reaches 0.1 and 0.2, that's when the variance spikes up dramatically to 26.08. At 26.08 (where it's bolded on the table), the variance spiked the highest, rightfully so due to the fact that the Bayesian A2C is under the highest attack. Unfortunately, when the noise attack is at 0.2, the performance on rewards does drop on the Bayesian A2C (and the deterministic A2C). However, the fortunate part and more important aspect is that the Bayesian model captures the uncertainty over the actions (variance) when the robot is under attack or in noisy conditions, especially in high attacks/noise. Notably, Table 4.2 shows how variance captures the model's behavior of the uncertainty over the actions when there is a very high attack on the robot and highlights the model's self-awareness when the robot is under attack and becomes aware when it's very uncertain about the actions. Thus, the Bayesian model is aware when it's in a noisy or high attack condition because the variance spikes up dramatically, and it becomes aware that the actions the model is predicting are very uncertain. There may be a better action to do or a better-exploring trajectory. The model predicts the best

possible case for the circumstance it's in but can only do so much considering that constant noise and attacks are being thrown at it, disturbing its view and environment. With the variance, this can raise a flag to itself or possibly a warning signal to a human user to make that human interaction adjustment whenever the model becomes very uncertain and is aware it's under high attack/noise. This mechanism of variance can be used for self-awareness on the robot and can be fed back to the robot to cause a self-warning mechanism to itself and into its algorithm/code to make proper adjustments.

Or, better yet, extend this to other robotic applications, such as biomedical, surgeries, patent diagnoses, automation, self-driving, drones, etc., where humans are involved, such as doctors, nurses, drivers, or other specialized experts who can interact with and adjust the behavior of the robot and its predictions when it is under attack or in high-noise conditions. Thus, when uncertainty spikes up about its predictions (variance), the robot becomes aware of this and sends a warning signal for caution or alert to itself and the human user to fix the issue the robot is f acing. This approach may reduce errors and improve precision in multiple applications that rely on robots to learn and predict various tasks by becoming aware when they are uncertain about their predictions due to the conditions they face, such as being under attack or in high noise interference.

In contrast, the Bayesian A2C model exhibits remarkable stability even in the presence of noisy environments. Throughout the epochs run in various noise conditions, the model's performance experienced only minor drops in the analyzed moving average and cumulative reward, demonstrating consistent strong robustness. These results indicate that the Bayesian A2C model excels in handling noisy environments, as evidenced by the consistently high reward values it achieves. This superior robustness of the proposed Bayesian A2C model against noisy environments can be attributed to its capability to propagate uncertainty through the layers and non-linearities of the Actor and Critic networks. This uncertainty is derived from the covariance matrix of the variational posterior distribution of the Actor and Critic network parameters. By incorporating this additional information in the form of a covariance matrix during navigation and exploration of the

environment during training, the learning process is enhanced, leading to a more noise-resistant model, as presented in our proposal. Hence, the improved robustness to noise.

CHAPTER V

CONCLUSION

This thesis presents an innovative approach to active simultaneous localization and mapping (SLAM) through the utilization of Bayesian A2C reinforcement learning. The central focus of this method is to effectively manage uncertainty through a network, particularly the A2C. This is accomplished by incorporating Bayesian inference techniques to integrate the mean and covariance values from the variational posterior distribution into both the Actor and Critic networks. Through the non-linear activation function using first-order Taylor series approximation between layers, it estimates uncertainty and predicts an action. The mean and covariance are propagated to estimate the predictive distribution of the robot's actions. Specifically, the mean serves as the robot's predicted actions, while the covariance provides valuable insights into the uncertainty associated with each action. Experiments were conducted within the OpenAI gym Gazebo simulator that demonstrated the substantial enhancement and robustness in navigation within noisy environments achieved by this newly proposed Bayesian A2C approach. This proposed technique demonstrated superior performance, remarkable results and stability within various noisy simulated scenarios interrupted by Gaussian and adversarial noise.

5.1 Future Works

This study could be extended to real-world robots in future work, with real-world distortions and noises introduced into the environment to analyze the model's robustness in these cases. The study could also be extended to other forms of robots, such as underwater robots, medical robots, micro-robots and drones, to examine how the model improves their movement in both simulation tests and real-world scenarios.

With this new proposed self-awareness mechanism, including uncertainty, a robot under high noise stress could raise a self-warning signal and adjust its behavior accordingly. Or, better yet, it could raise a warning signal to a human user, who could then intervene to solve the problem. For example, in the biomedical field, if an agent becomes very uncertain about the predictions it is choosing, it could raise a warning signal to a doctor or nurse, who could then make the necessary human adjustment on a particular patient. Another robotic application that could be improved is autonomous driving, where the robot may be uncertain about certain situations and could raise a signal to the driver so that they can maneuver or make the necessary adjustments. Many other robot applications could be discussed and explained, such as air drones, search-and-rescue robots, submarines, space applications, and automation. In all of these fields, it is important to have a strong and robust action plan in place when conditions and environments change unexpectedly.

Having a self-warning signal or an alert signal to a human user to make the adjustment outside the robot's domain is crucial because this new method can definitely reduce the risk of human error, improve precision in diagnosing patients, assist nurses/doctors/drivers/itself, etc., and improve the robot's performance in whatever its environment and goals are. By introducing uncertainty into the robot and allowing it to have more self-awareness about its environment, the changes in the environment, the uncertainty on the choices it is predicting based on the changing conditions, and the system's behavior, this research and implementation can be extended to a variety of other robotic applications where robust actions are critical and minimal to zero error is needed.

REFERENCES

[1] M. F. AHMED, K. MASOOD, AND V. FREMONT, *Active SLAM: A review on last decade*, 2023.

[2] M. ALCALDE, M. FERREIRA, P. GONZÁLEZ, F. ANDRADE, AND G. TEJERA, *DA-SLAM: Deep active SLAM based on deep reinforcement learning*, in 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE), 2022, pp. 282–287.

[3] D. M. BLEI, A. KUCUKELBIR, AND J. D. MCAULIFFE, *Variational inference: A review for statisticians*, Journal of the American Statistical Association, 112 (2017), pp. 859–877.

[4] N. BOTTEGHI, B. SIRMAÇEK, M. POEL, AND C. BRUNE, *Reinforcement learning helps SLAM: Learning to build maps*, ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 43 (2020), pp. 329–336.

[5] S. L. BRUNTON AND J. N. KUTZ, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, Cambridge University Press, 2019.

[6] H. CARRILLO, I. D. REID, AND J. A. CASTELLANOS, *On the comparison of uncertainty criteria for active SLAM*, 2012 IEEE International Conference on Robotics and Automation, (2012), pp. 2080–2087.

[7] D. DERA, N. C. BOUAYNAYA, G. RASOOL, R. SHTERENBERG, AND H. M. FATHALLAH-SHAYKH, *Premium-cnn: Propagating uncertainty towards robust convolutional neural networks*, IEEE Transactions on Signal Processing, 69 (2021), pp. 4669–4684.

[8] B. FITELSON, *Studies in bayesian confirmation theory*, UNIVERSITY OF WISCONSIN, 2001.

[9] T. HAARNOJA, A. ZHOU, P. ABBEEL, AND S. LEVINE, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, 2018.

[10] M. HESSEL, J. MODAYIL, H. VAN HASSELT, T. SCHAUL, G. OSTROVSKI, W. DABNEY, D. HORGAN, B. PIOT, M. AZAR, AND D. SILVER, *Rainbow: Combining improvements in deep reinforcement learning*, 2017.

[11] B. JANG, M. KIM, G. HARERIMANA, AND J. W. KIM, *Q-learning algorithms: A comprehensive classification and applications*, IEEE Access, 7 (2019), pp. 133653–133667.

[12] S. KARAGIANNAKOS, *The idea behind actor-critics and how a2c and a3c improve them*, Nov 2018.

[13] S. MACENSKI, T. FOOTE, B. GERKEY, C. LALANCETTE, AND W. WOODALL, *Robot operating system 2: Design, architecture, and uses in the wild*, Science Robotics, 7 (2022).

[14] S. M. MAKAM AND P. GOPAL, *Active SLAM*, Sep 2020.

[15] V. MNIH, A. P. BADIA, M. MIRZA, A. GRAVES, T. P. LILLICRAP, T. HARLEY, D. SILVER, AND K. KAVUKCUOGLU, *Asynchronous methods for deep reinforcement learning*, 2016.

[16] V. MNIH, K. KAVUKCUOGLU, D. SILVER, A. A. RUSU, J. VENESS, M. G. BELLEMARE, A. GRAVES, M. A. RIEDMILLER, A. K. FIDJELAND, G. OSTROVSKI, S. PETERSEN, C. BEATTIE, A. SADIK, I. ANTONOGLOU, H. KING, D. KUMARAN, D. WIERSTRA, S. LEGG, AND D. HASSABIS, *Human-level control through deep reinforcement learning*, Nature, 518 (2015), pp. 529–533.

[17] E. F. MORALES, R. MURRIETA-CID, I. BECERRA, AND M. A. ESQUIVEL-BASALDUA, *A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning*, Intelligent Service Robotics, 14 (2021), pp. 773 – 805.

[18] A. PLAAT, *Deep Reinforcement Learning*, Springer Nature Singapore, 2022.

[19] J. A. PLACED AND J. A. CASTELLANOS, *A deep reinforcement learning approach for active SLAM*, Applied Sciences, (2020).

[20] J. A. PLACED AND J. A. CASTELLANOS, *A general relationship between optimality criteria and connectivity indices for active graph-SLAM*, 2022.

[21] J. A. PLACED, J. STRADER, H. CARRILLO, N. ATANASOV, V. INDELMAN, L. CARLONE, AND J. A. CASTELLANOS, *A survey on active simultaneous localization and mapping: State of the art and new frontiers*, 2023.

[22] F. PUKELSHEIM, *Optimal Design of Experiments*, Society for Industrial and Applied Mathematics, 2006.

[23] M. L. RODRÍGUEZ-ARÉVALO, J. NEIRA, AND J. A. CASTELLANOS, *On the importance of uncertainty representation in active SLAM*, IEEE Transactions on Robotics, 34 (2018), pp. 829–834.

[24] J. SCHULMAN, F. WOLSKI, P. DHARIWAL, A. RADFORD, AND O. KLIMOV, *Proximal policy optimization algorithms*, 2017.

[25] H. SURMANN, C. JESTEL, R. MARCHEL, F. MUSBERG, H. ELHADJ, AND M. ARDANI, *Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments*, 2020.

[26] R. S. SUTTON, F. BACH, AND A. G. BARTO, *Reinforcement Learning: An Introduction Policy Approximation Actor-Critic Methods*, A Bradford Book, 2nd ed., 2018.

[27] ——, *Reinforcement learning: An introduction/Introduction/An Extended Example: Tic-Tac-Toe*, A Bradford Book, 2nd ed., 2018, p. 10–15.

[28] ——, *Reinforcement learning: An introduction/The Reinforcement Learning Problem/Goals and Rewards*, A Bradford Book, 2nd ed., 2018, p. 56–57.

[29] R. TANGRI, *Approximating kl divergence*, May 2021.

[30] H. VAN HASSELT, A. GUEZ, AND D. SILVER, *Deep reinforcement learning with double q-learning*, 2015.

[31] A. VIOLANTE, *Simple reinforcement learning: Temporal difference learning*, Oct 2018.

[32] Z. WANG, T. SCHAUL, M. HESSEL, H. VAN HASSELT, M. LANCTOT, AND N. DE FREITAS, *Dueling network architectures for deep reinforcement learning*, 2016.

[33] S. WEN, Z. JI, A. B. RAD, AND Z. GUO, *A hybrid technique for active SLAM based on rppo model with transfer learning*, Jan 2022.

[34] S. WEN, Y. ZHAO, X. YUAN, Z. WANG, D. ZHANG, AND L. MANFREDI, *Path planning for active SLAM based on deep reinforcement learning under unknown environments*, Intelligent Service Robotics, 13 (2020), pp. 263–272.

[35] V. ZAMBALDI, D. RAPOSO, A. SANTORO, V. BAPST, Y. LI, I. BABUSCHKIN, K. TUYLS, D. REICHERT, T. LILLICRAP, E. LOCKHART, M. SHANAHAN, V. LANGSTON, R. PASCANU, M. BOTVINICK, O. VINYALS, AND P. BATTAGLIA, *Relational deep reinforcement learning*, 2018.

[36] I. ZAMORA, N. G. LOPEZ, V. M. VILCHES, AND A. H. CORDERO, *Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo*, 2017.

BIOGRAPHICAL SKETCH

Bryan J. Pedraza was born and raised in McAllen, Texas. He graduated in the top ten percent of his class from PSJA Memorial High School in 2016 and went on to earn a Bachelor of Science in Computer Engineering with summa cum laude honors from the University of Texas at San Antonio (UTSA) in May 2021. During his undergraduate studies, for two years, he worked as an IT Technician in the Biomedical and Industrial department and participated in the IEEE program.

Pedraza earned his Master of Science in Electrical Engineering degree from the University of Texas Rio Grande Valley at Edinburg, Texas in December 2023. His research interests included artificial intelligence (AI), machine learning, deep learning, reinforcement learning, robotics, and computer vision processing, which formed the core foundation of his thesis work and provided him with the necessary tools to make a significant and original contribution to his field of study.

For one and a half years of his graduate program, he participated in the GAANN fellowship program and served as a Graduate Teaching Assistant for several courses, teaching students the basics of lab equipment and circuit analysis. He also submitted a paper and presented his research work at the IEEE Conference on Artificial Intelligence (CAI) in Santa Clara, California, on June 5-6, 2023.

Pedraza's thesis work is a testament to his dedication to electrical engineering and his ambition to extend his knowledge and advance his career in this field. He invested countless hours researching, analyzing, and applying critical thinking to better understand the functioning of machine learning, robotics, AI, neurons, uncertainty, and other concepts. Pedraza remains committed to continuing to make valuable contributions to the field of electrical engineering and machine learning. To contact him personally, please use the following email address: bryanjp53@gmail.com.