TODO: TITLE

TODO: AUTHOR NAMES

ABSTRACT.

1. INTRODUCTION

1.1. **Contributions.**

2. BACKGROUND

Q-learning is an algorithm for reinforcement learning that aims to learn the optimal action given a state. Q-learning takes into account the state, actions, and rewards to learn a policy that maximizes the cumulative reward over time.

The Bellman equation is the fundamental equation in reinforcement learning and outputs the cost of taking an action in a given state, then following the optimal policy thereafter. The Bellman equation is defined as:

$$Q(s,a) = R(s,a) + \gamma \max_{a'} Q(s',a') \tag{1}$$

The optimal policy is the policy that maximizes the cumulative reward, and finding this optimal policy is the goal of reinforcement learning.

Generally, the Q-values are stored in a table, calculated using a function approximator, or found using a Markov chain, but this limits the complexity of the environments that can be solved.

Deep Q-Networks (DQN) provided a breakthrough in reinforcement learning through the process training a deep neural network to approximate the Q-values, allowing for reinforcement learning to be applied to much more complex environments, a previous limitation of traditional Q-learning.

Some difficulties with previous Q-learning methods were instability and divergence when a non-linear function approximator was used.

DQN utilizes experience replay to provide stability during training by storing the experiences of the agent, and batch sampling from this storage to break up correlations in the observation sequence.

3. RELATED WORK

4. METHOD

For each of the following models, the forward pass takes as input a stack of 4 grayscale frames of size 84x84 pixels, and outputs Q-values for each possible action in the environment.

Training each model...

4.1. **Baseline Model.**

The baseline model is a DQN based on the architecture proposed by Mnih et al. in "Playing Atari with Deep Reinforcement Learning". The model consists of a convolutional neural network (CNN), intaking 4 stacked frames of 84x84 pixels and utilizing three convolutional layers (32 filters of size 8x8 with stride 4, 64 filters of size 4x4 with stride 2, and 64 filters of size 3x3 with stride 1), followed by two fully connected layers (Input size 3136 to 512 units, and 512 units to output layer with number of actions). The ReLU activation function is applied after each convolutional and fully connected layer except for the output layer.

4.2. **Dueling DQN Variant.**

The dueling DQN is identical to the original baseline model up until the final convolutional layer. After this layer, there is a hidden layer with 512 units and a RELU activation function. This gets fed into the Value stream and the Advantage stream (nn.Linear(512, 1) and nn.Linear(512, num$_a ctions)respectively).Theoutputsofthesetwostreamsar$
$valuesusingtheequationQ(s,a) = V(s) + (A(s,a)mean_a A(s,a))$.

The purpose of this architecture is such that the model learns the value of a state separately from an action's advantage, which can help in scenarios where many actions have similar outcomes.