PLAYING ATARI WITH DEEP REINFORCEMENT LEARNING

START!

Ali Muhammad Asad

Dua Batool

ARE YOU READY?

INTRODUCTION

- First Deep Learning model to successfully control policies directly from high dimensional sensory input
- Model is a convolutional neural network, trained with a variant of Q-Learning
- Applied to 7 Atari 2600 games from Arcade Learning Environment, with no adjustment of the architecture or learning algorithm
- Model out-performs all previous approaches on 6 of the games, and surpasses human experts on 3 of them
- Pong, Breakout, Space Invaders, Seaquest, Beam Rider


2
8


154270
OXYGEN
ACTIVISION
By DePaula


ATARI
ATARI
ATARI


SECTOR 01
11

THE PROBLEM

PROBLEM STATEMENT

- **Challenge**: Learning control policies directly from high-dimensional sensory inputs like vision and speech poses a longstanding challenge in reinforcement learning (RL).
- **Current Approach Limitations**: Successful RL applications rely on hand-crafted features, hindering scalability and generalization.
- **Deep Learning Potential**: Recent advancements in deep learning offer promise in extracting meaningful features from raw sensory data.
- **Deep RL Challenges**: However, applying deep learning to RL faces hurdles like sparse, noisy rewards, correlated states, and non-stationary distributions.

MOTIVATION

Unlocking Deep Learning's Power

Innovative Approach

Real-world Application

SOLUTION

FORMULATION

Leveraging Deep Learning

Inspiration from TD-Gammon

Incorporating Experience Replay

Addressing Computational Challenges

Deep Q-Learning Algorithm

RL AGENT

- Agent acts with the Atari emulator in a sequence of actions, observations and rewards
- Selects an action from the set of legal game actions at each time-step
- Receives a reward or penalty representing the change in game score
- Considers sequences of actions and observations, learning game strategies depending on those sequences
- The goal of the agent is to interact with the emulator by selecting actions to maximise the future reward
- Rewards discounted by a factor γ per time step
- All sequences expected to terminate - game has to end - gives rise to a large but finite Markov Decision Process

DEEP Q-LEARNING

- Use experience replay; agent’s experience “e” at each time step is stored in a data set D pooled over many episodes into replay memory
- Apply Q-learning updates / mini-batch updates to samples of experience after drawing “e ~ D” random from the pool
- After experience replay, agent selects and executes an action according to an ε-greedy policy
- Q-function works on a fixed length represnetaiton of histories produced by a function φ, instead of histories of arbitrary length as inputs to a neural network

GAME PLAY (ALG)

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

WHY THiS APPROACH?

- Each step of solution potentially used in many weight updates - greater data efficiency
- Randomizing sample breaks correlations between consecutive samples - reduces variations of the updates
- When learning on-policy, current parameters determine the next data sample that the parameters are trained on
- Experience replay causes behaviour distribution to be averaged over many previous states, smoothing out learning and avoiding divergence in parameters

REWARD

Positive rewards are fixed to 1, negative rewards to -1, and 0 rewards remain unchanged during training. This standardization of rewards aims to facilitate learning by limiting the scale of error derivatives.

The scale of scores varies greatly from game to game, and standardizing rewards simplifies the training process.

VALUE FUNCTION

Q-function estimates the expected cumulative reward from a given state-action pair. The Q-function is updated iteratively based on observed rewards and transitions.

Deep Q-learning algorithm combines experience replay with Q-learning updates to enhance data efficiency and stability in estimating the value function.

POLICY

agent's action selection mechanism based on its current state

POLICY EXPLORATION

Exploration-exploitation dilemma: the agent needs to balance between exploring new actions to discover optimal strategies and exploiting known strategies to maximize rewards.

Use of an ε-greedy policy during training, where with probability ε, the agent selects a random action to explore, and with probability 1-ε, it selects the action with the highest estimated Q-value to exploit known strategies.

POLICY EVALUATION

Policy is evaluated using metrics such as average total reward obtained in episodes or games, as well as the estimated action-value function Q.

LEARNING FROM POLICY

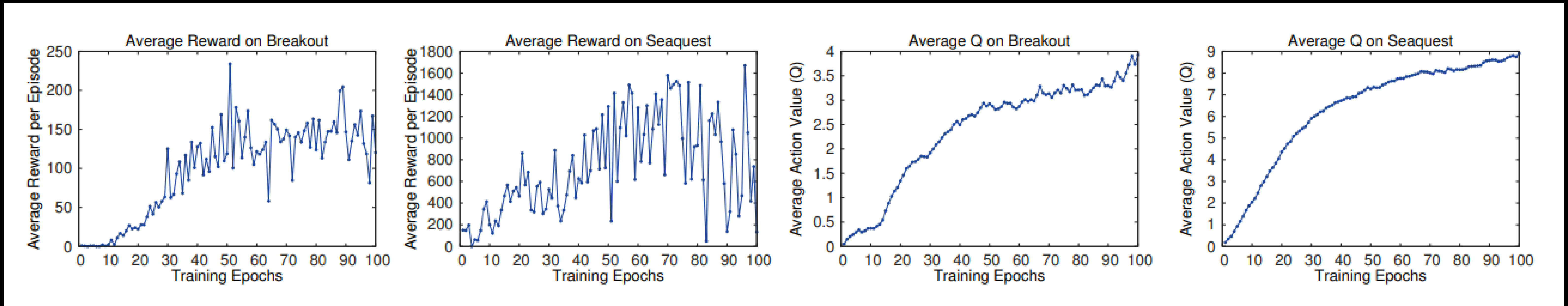
- The Deep Q-learning algorithm updates the parameters of the Q-function based on experiences sampled from the agent's interactions with the environment.
- By iteratively updating the Q-function, the algorithm learns an optimal policy that maximizes expected cumulative rewards over time.

EXPERIMENTS / RESULTS

GAME SETTINGS

- Beam Rider, Breakout, Enduro, Pong, QBert, Seaquest, Space Invaders
- Same network architecture, learning alg, and hyperparameter settings across all games - robust approach
- +ve rewards +1, -ve rewards -1, unchanged rewards 0
- minibatches of size 32
- ε-greedy policy used in training
- Training done on 10 million frames
- replay memory of 1 million most recent frames

GAME SCORE (RESULTS)



Average reward per episode on the left,
Average maximum predicted action value on the right,

30 minutes training time

Seaquent and Breakout

HALL OF FAME (HiSCORE)

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Average total reward values for various learning methods on an ε-greedy policy

CONCLUSION

- The novel deep learning model for reinforcement learning achieves state-of-the-art performance in Atari games, demonstrating the power of direct learning from raw pixels.
- By combining online Q-learning with experience replay, efficient training of deep networks in RL tasks is introduced.
- This success opens doors for further exploration and application of deep reinforcement learning in diverse real-world scenarios.

THANK YOU

END