Playing FPS Games

- Siddharth Nahar (2016CSB1043)
- Sameer Arora (2016csb1058)
- Prasad Kshirsagar (2016CSB1041)
- Girish Kumar (2016CSB1040)

Problem Definition

- **Statement** : Make AI play FPS games with reinforcement learning under different scenarios like deathmatch, Protecting the center, Multi-Deathmatch etc.

Central Objectives

- Understanding & applying the basic concepts of deep Q-learning (deep reinforcement learning).
- Configuring the Vizdoom API : customize difficult game scenarios to train our models.

Available Literature

- DeepMind [1] Playing Atari with Deep Reinforcement Learning (DQN).
- Thomas Simonani [2] DQN with experience replay, and fixed Q-targets.
- Matthew Hausknecht, Peter Stone [3] Deep Recurrent Q-Network.
- ViZDoom [4] Doom-Based AI research platform.

Reinforcement Learning

- There is no supervisor, only a reward signal
- sequential, non i.i.d data
- All goals can be described by the maximisation of expected cumulative reward

An RL agent may include one or more of these components:

- Policy: agent’s behaviour function (Deterministic/Stochastic)
- Value function: how good is each state and/or action (Expected Future Reward)

1. Predict future Reward.
2. Used Bellman Equation for predicting Reward used to select action to perform.

$$Q(s,a) = r(s,a) + \underline{\gamma max_a Q(s',a)}$$

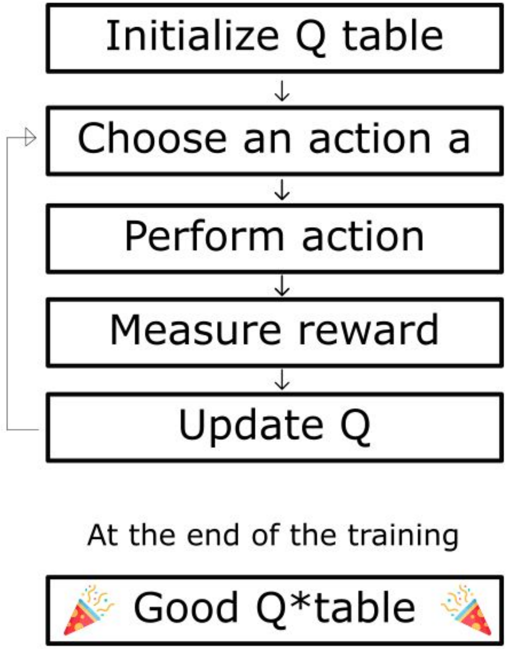
Q-Learning

Agent tries to learn optimal policy from its history of interaction with environment.History is a sequence of experiences tuples of (state,action,reward,next_state)

Q-learning uses temporal differences to estimate the value of Q*(s,a) to create and update a Q-table a matrix of states and actions.

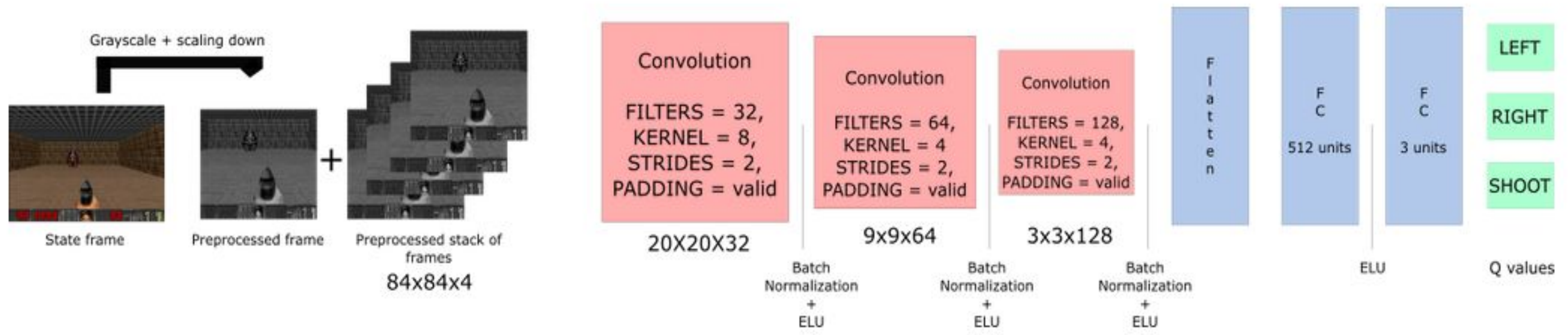
$$\underbrace{NewQ(s,a)}_{\text{New Q value for that state and that action}} = \underbrace{Q(s,a)}_{\text{Current Q value}} + \underbrace{\alpha}_{\text{Learning Rate}}[\underbrace{R(s,a)}_{\text{Reward for taking that action at that state}} + \underbrace{\gamma}_{\text{Discount rate}} \underbrace{max Q'(s',a')}_{\text{Maximum expected future reward given the new s' and all possible actions at that new state}} - Q(s,a)]$$

The Q-learning algorithm Process



Deep Q-Learning

Q-table is not feasible for millions of states and action pairs as in most of FPS games.Thus,we use some Neural networks to as function approximator of Q*().



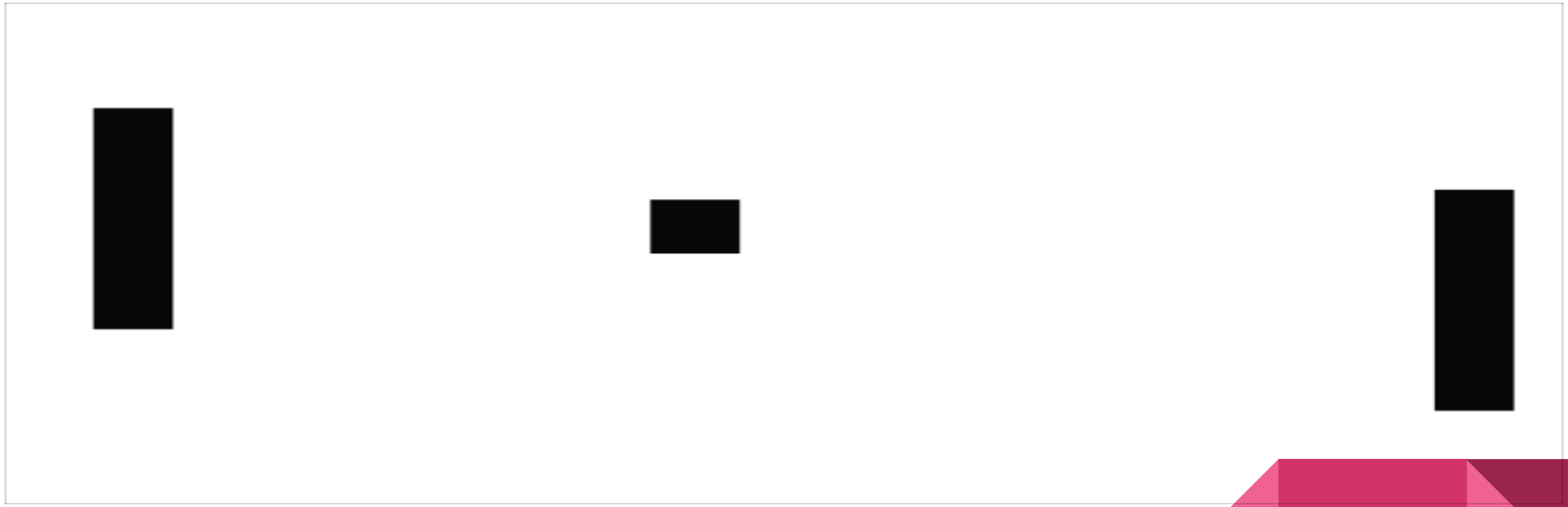
Pre-processing

Pre-processing a frame :

- Grayscale each of our frames
- Crop the screen
- We normalize pixel values
- Finally we resize the preprocessed frame

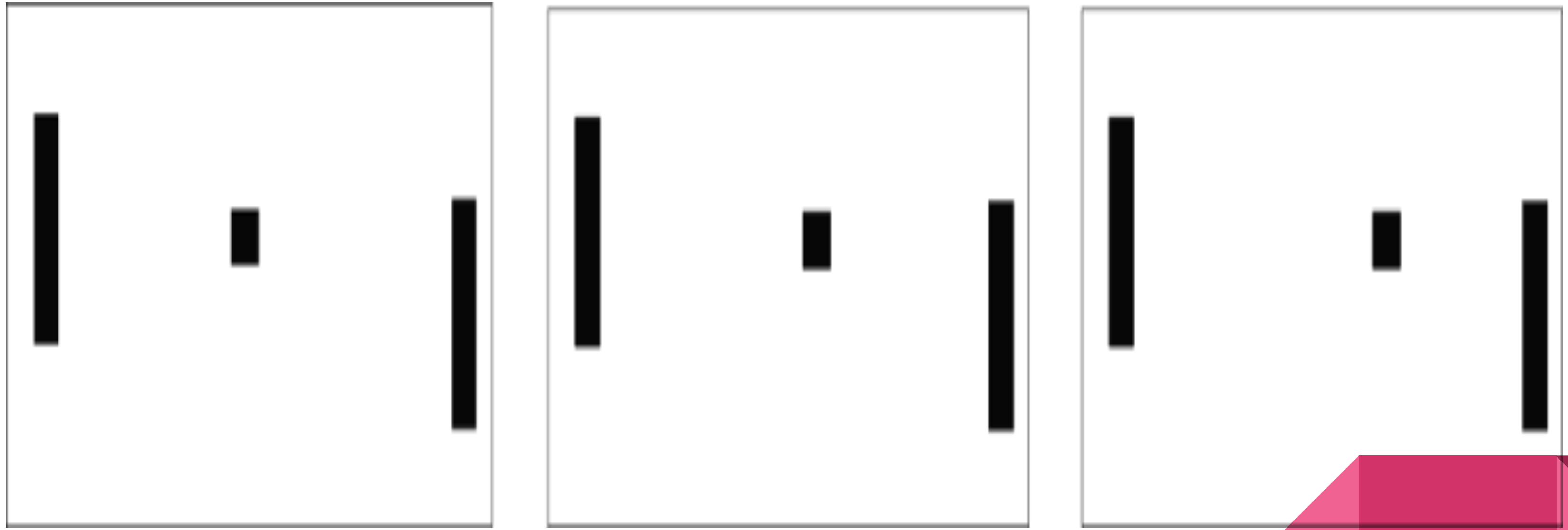
Problem of Temporal Limitation

Single Frame (for example) :

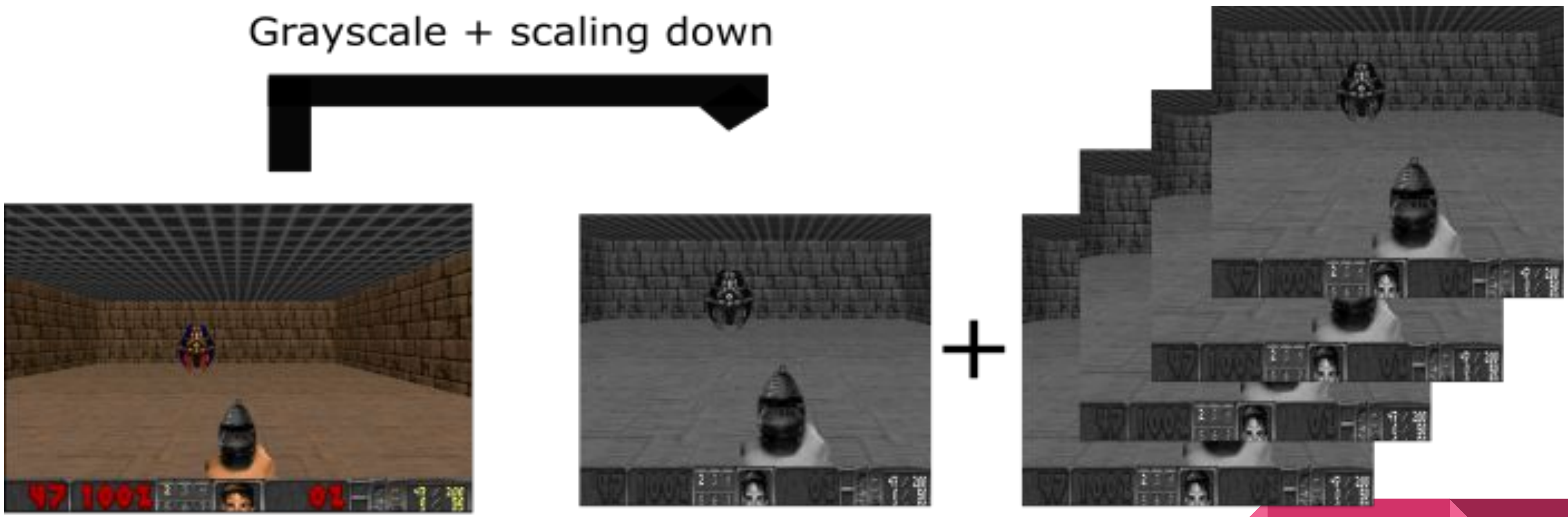


Stacking Frames

Snapshot of sequential frames (of example) :



Final demo after pre-processing



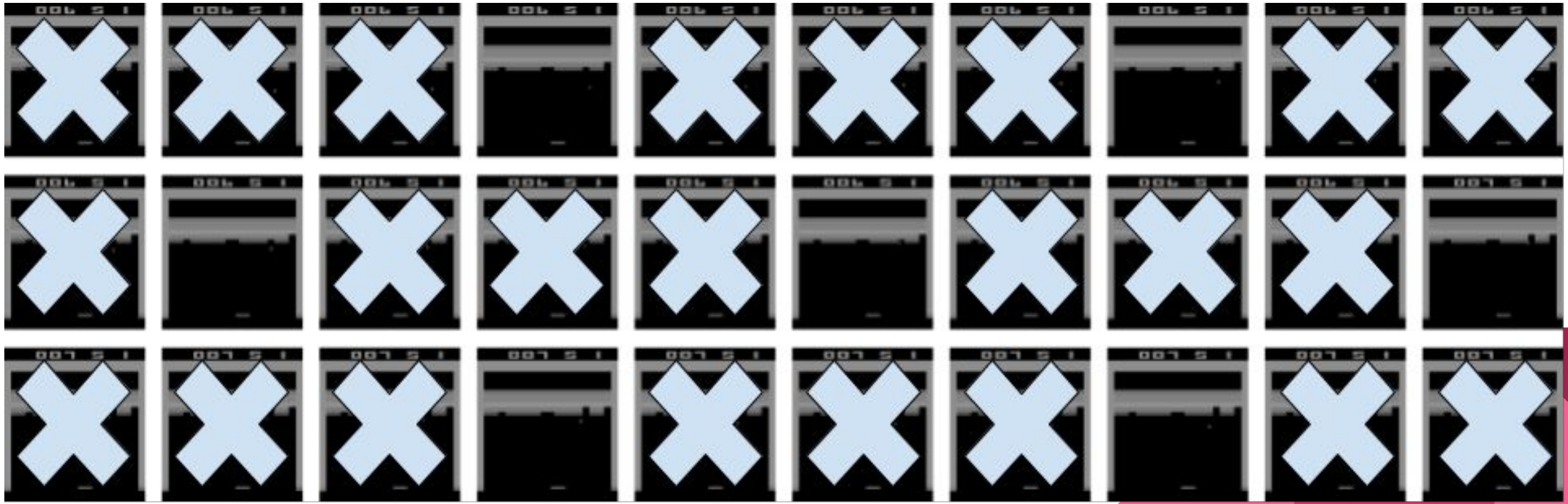
State frame

Preprocessed frame

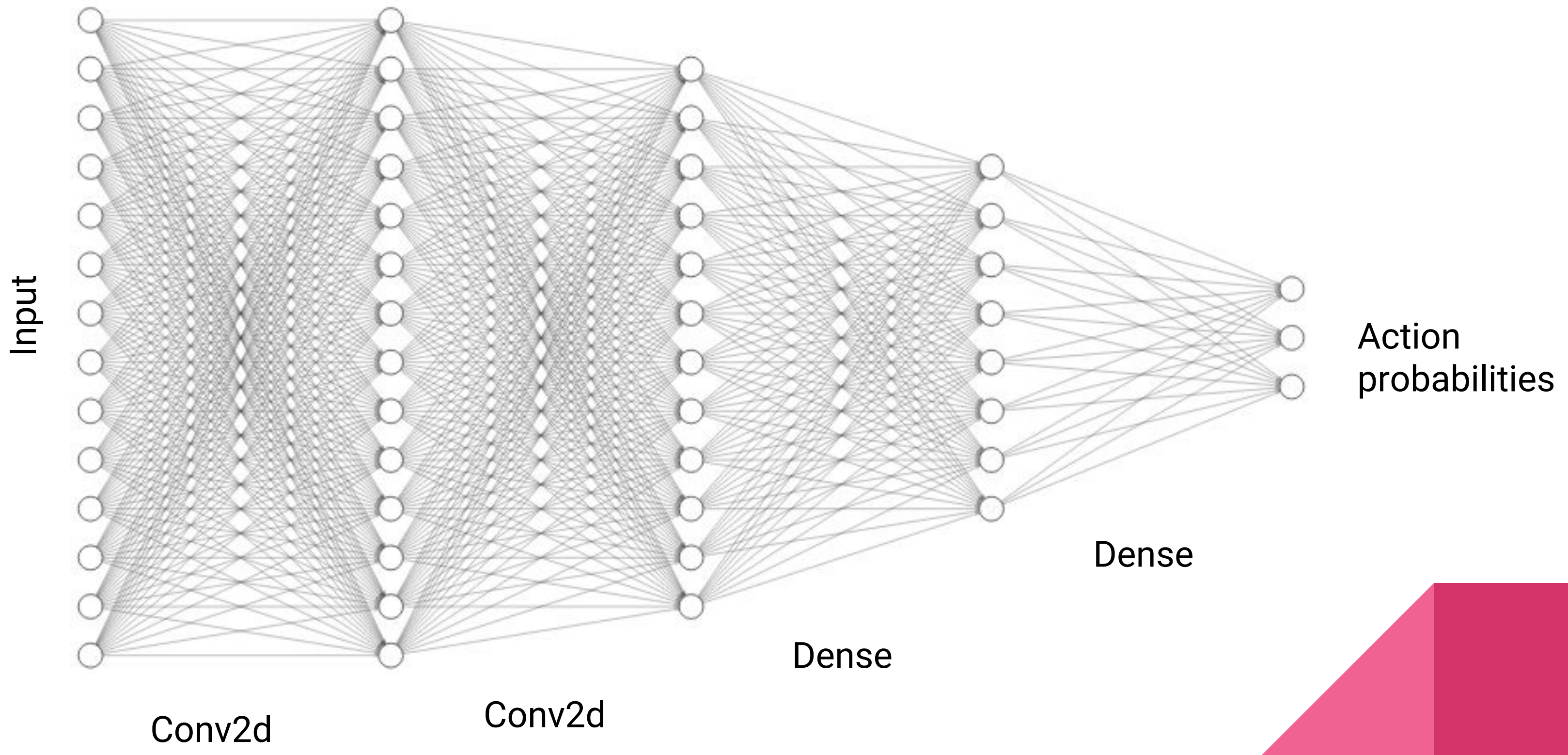
Preprocessed stack of frames

Skipping Frames

- Disadvantage of stacking consecutive frames ?
- **Solution** : Skip in-between frames
- Skipping frames has proved to be efficient in decision making.



Model Design



What layers are doing?

- Input layer contains frames from memory buffer.
- First two convolution layers are getting features from state frames.
- Other two dense layers are learning complex rules based on the features.
- Probabilities of the actions are returned at the output layer for the scenario.

Experience Replay

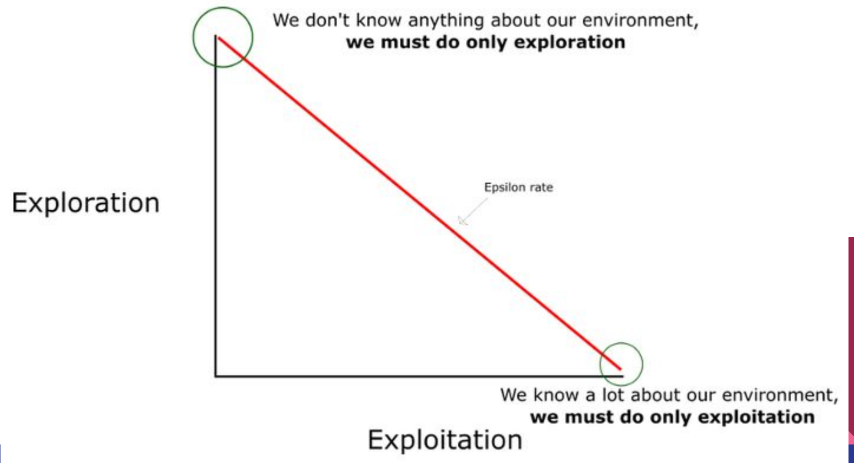
- Why we need previous experiences ?
  - Avoid forgetting previous experiences.
  - Reduce correlations between experiences.

1. While Receiving Sequential data we tend to forget previous experience this will lead to loss of information about those experiences.
2. Due to receiving similar data of sequential frames there is high correlation between training data So we keep memory buffer to select states randomly from memory which consists history of all experiences.
3. Use a Replay Memory to store various experiences in a buffer and sample from the buffer during training.

Exploration/Exploitation Trade-Off

- We specify an exploration rate “epsilon,” which we set to 1 in the beginning. This is the rate of steps that we’ll do randomly. In the beginning .Implying need to do a lot of exploration, by randomly choosing our actions.
- We generate a random number between 0 and 1. If this number > epsilon, then we will do “exploitation” (We select the best action at each step as learned till now). Else, we’ll do exploration.

- We reduce it progressively as the agent becomes more confident at estimating Q-values.


We don't know anything about our environment, **we must do only exploration**
Exploration
Epsilon rate
We know a lot about our environment, **we must do only exploitation**
Exploitation

Training Model(Putting All Together)

1. Initialize the Game Environment,Replay Memory and the Neural Network for learning.
2. For each epoch run a episode of game with maximum limit of steps possible per episode.
3. Each step choose among exploration/exploitation using previous strategy.
4. Add the experiences to the Replay Memory and then sample from the memory to perform training.
5. Test the model for some new game episodes and save the model.

Results

1)Basic Scenario : The purpose of this scenario is to teach the agent how to KILL enemy without knowing Its position.

Rewards - +101 for killing the monster -5 for missing Episode ends after killing the monster or on timeout, living reward = -1

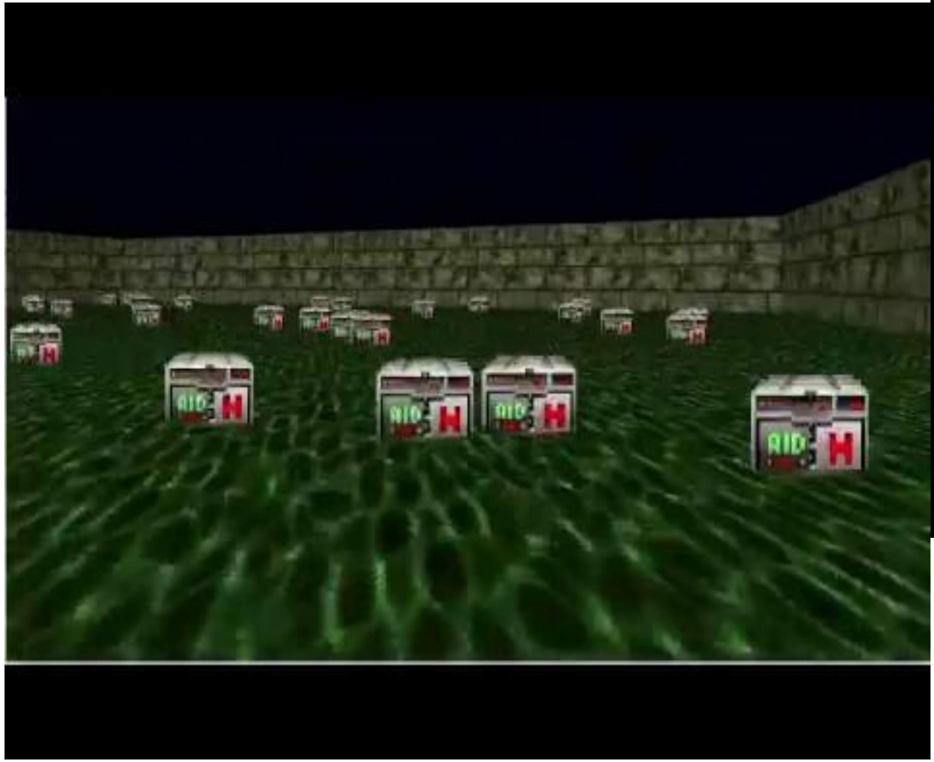
3 available buttons: move left, move right, shoot (attack) , timeout = 300

2)HEALTH GATHERING : The purpose of this scenario is to teach the agent how to survive without knowing what makes him survive.

Rewards - living_reward = 1,death penalty = -100

3 available buttons: turn left, turn right, move forward
1 available game variable: HEALTH

Shoot and Kill Scenario ⟶



⟵ Health Gathering

Future Work

1. Apply new techniques developed in Deep Reinforcement Learning like Fixed Q-Target, DDQN(Dueling Double QN) , Prioritized Experience Replay , Deep Recurrent QN.
2. Develop a more complex scenario to train our agent like in multi-bots, DeathMatch, Protect the Centre.

REFERENCES

1. DeepMind “Playing Atari with Deep Reinforcement Learning”, 2013
2. Thomas Simonani “Improvements in Deep Q Learning: Dueling Double DQN, Prioritized Experience Replay, and fixed Q-targets”
3. Matthew Hausknecht, Peter Stone “Deep Recurrent Q-Learning for Partially Observable MDPs”, 2015
4. VizDoom “A Doom-based AI Research Platform for Visual Reinforcement Learning”

Thank you