A

Major Project

On

AUTOMATING PACMAN WITH DEEP Q-LEARNING USING PYGAME

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

N.SAI KUMAR (187R1A05A1)

SK.AFROZ (187R1A05B5)

U.VARSHA (187R1A0572)

Under the Guidance of

G.VIJAY KUMAR

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICALCAMPS UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2018-22

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**AUTOMATING PACMAN WITH DEEP Q-LEARNING USING PYGAME"** being submitted by **N.SAI KUMAR (187R1A05A1),SK.AFROZ(187R1A05B5) & U.VARSHA(187R1A0572)** in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of Bonafede work carried out by him/her under our guidance and supervision during the year 2021-22.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

G.Vijay Kumar
Assistant Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K.Srujan Raju
HOD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on _____

ACKNOWLEGDEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We take this opportunity to express my profound gratitude and deep regard to my guide **G.Vijay Kumar,** Assistant Professor for his exemplary guidance, monitoring and constant encouragement through out the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) **Mr. J. Narasimha Rao, Dr. T. S. Mastan Rao, Mr. A. Uday Kiran, Mr. A. Kiran Kumar, Mrs. G. Latha** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We also express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help. Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

N.SAIKUMAR(187R1A05A1)

SK.AFROZ (187R1A05B5)

U.VARSHA (187R1A0572)

ABSTRACT

We apply various reinforcement learning methods on the classical game. Pacman, we study and compare Q-learning, approximate Q-learning and Deep Q-learning based on the total rewards and win-rate, While Q-learning has been proved to be quite effective on smallGrid, it becomes inefficient to find the optimal policy in large grid-layouts, In approximate Q-learning, we handcraft ‘intelligent’ features to feed into the game. The main purpose of this project is to investigate the effectiveness of Deep Q-learning based on the context of the Pacman game having Q-learning and Approximate Q-learning as baselines. Reinforcement learning creates its own, ever-shifting dataset, both because the network generates its own target values and because the action choices of the network directly impact which states it will reach in its environment and therefore what it will have to learn about.

LIST OF FIGURES

TABLE OF CONTENTS

1.INTRODUCTION

1.INTRODUCTION

1.1 PROJECT SCOPE

Machine learning has become a popular research area, combining all sorts of algorithms for all kinds of tasks. But what they have in common no matter the task is that all machine learning tasks require some sort of training data. This means that with every new parameter in the environment trained upon, the set of states in the environment grows exponentially. The pacman will first train automatically and play the game with wins and losses. After completion of the training it will play efficiently.

1.2 PROJECT PURPOSE

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. We have to create a small machine learning application and show the how to perform that.

1.3 PROJECT FEATURES

Reinforcement learning originated from the idea of learning something by interacting with its environment. But before such an experiment can even designed, the task experimented on has to full the Markov Property. We define our Deep Q-learning neural network**.** This is a CNN that takes in-game screen images and outputs the probabilities of each of the actions, or Q-values, in the Ms-Pacman game space. To acquire a tensor of probabilities, we do not include any activation function in our final layer.

2.SYSTEM ANALYSIS

2.SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to become.

2.1 PROBLEM DEFINITION

The automating pacman is an AI application in that we have to train the model for playing automatically. In which we are using the reinforcement algorithm and the deep q neural networks. But the actual problem is it takes more time for training and learning the basics of the game.

2.2 EXISTING SYSTEM

The Player guides the titular pacman through a maze to eat all pacman dots in the maze .when all the dots are eaten pacman is taken to the next stage of the game.Four ghosts, Blinky, Pinky, Inky and Clyde roam the maze, trying to catch Pac-Man- if a ghost touches him, a life is lost. When all lives have been lost, the game ends. Near the corners of the maze are four larger, flashing dots known as Power Pellets, provide Pac-Man with the temporary ability to eat the ghosts. The ghosts turn deep blue, reverse direction, and move slower when Pac-Man eats one. When a ghost is eaten, its eyes return to the ghost home where it is regenerated in its normal color. Blue ghosts flash white before they become dangerous again.The amount of time the ghosts remain vulnerable varies from one round to the next, but the time period generally becomes shorter as the game progresses. In later stages, the ghosts do not change colors at all, but they still reverse direction when a power pellet is eaten.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

- Overfitting of the data may occur.
- Accuracy is low due to the number of testcases.
- It takes many number of testcases for AI to learn playing pacman .

2.3 PROPOSED SYSTEM

DeepMind published the first version of its Deep Q-Network (DQN), a computer program capable of human-level performance on a number of classic Atari 2600 games. Just like a human, the algorithm played based on its vision of the screen. Starting from scratch, it discovered gameplay strategies that let it meet (and in many cases, exceed) human benchmarks. In the years since, researchers have made a number of improvements that super-charge performance and solve games faster than ever before. We've been working to implement these advancements in Keras — the open source, highly accessible machine learning framework — and in this post, we'll walk through the details of how they work and how they can be used to master Ms. Pac-man.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

- AI based user interface.
- One of the major advantages is without human interaction it will play automatically.

2.4 FEASIBILITY STUDY

The aim of the Automating pacman with deep Q-learning using pygame project is the how the machine learning application runs in a real world. First we have to train the model and that model learn and perform the action. In order to give the reader a better understanding of the problem ahead and how to accomplish the task of building an self-learning agent, this thesis offers the basic knowledge needed to understand the principles of reinforcement learning and deep learning used, the implemented features, the architecture built and the results of experiments that helped further enhance the network performance.

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development**.**

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system..

2.4.3 BEHAVIOURAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?

- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioural aspects are considered carefully and conclude that the project is behaviourally feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- System : Intel Processor i3/i5
- OS : Windows 7,8,10
- Input Devices : Keyboard, Mouse
- Ram : 4GB

2.5.2 SOFTWARE REQUIREMENTS

The following are some software requirements.

- Operating system : Windows 7,8,10
- Coding Language : Python
- Tool : Visual Studio code(VS),Anaconda
- Modules : tensorflow,gymAI,pygame

3.ARCHITECTURE

3.ARCHITECTURE

3.1 PROJECT ARCHITECTURE

This project architecture shows the procedure followed for how the process of execute the game.

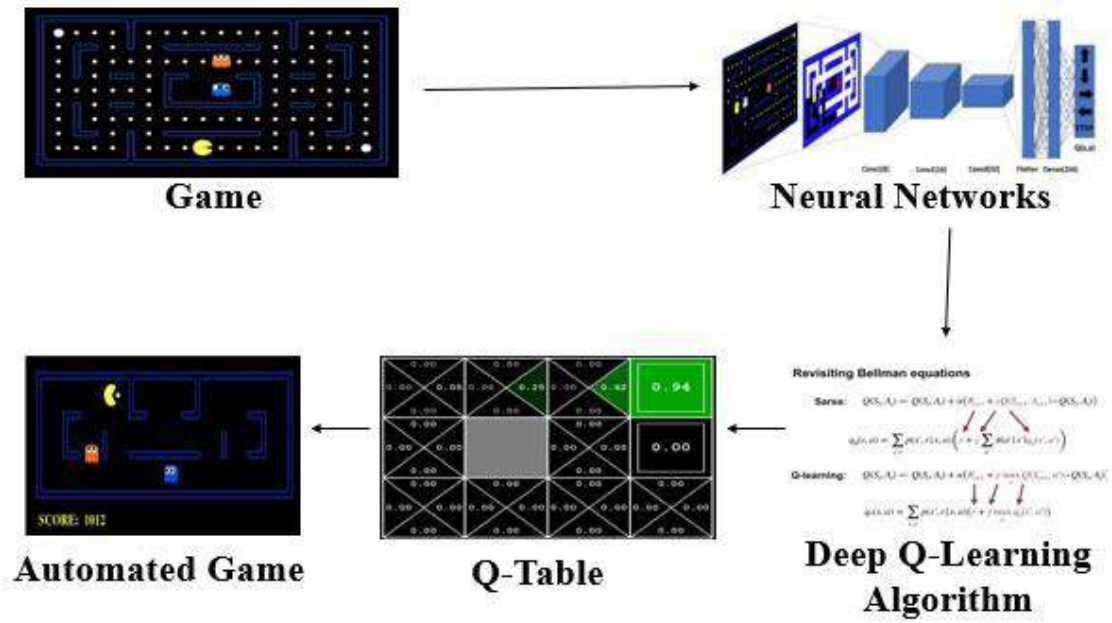

Figure 3.1: Project Architecture of Automating pacman game

DESCRIPTION

The above diagram represents the architecture of pacman game.

The Pacman framework of the UC Berkeley's introductory artificial intelligence course, CS 188 is used as a foundation for this experiment. For it a framework that has a good setting for learning the basics of reinforcement learning. It has a predefined environment, with states that hold all relevant game data and a pre-deend reward signal (,though it is not in-use in these experiment). And an agent interface, that makes it easy to implement new agents, just needing the user to keep some name conventions on the agent's name and the agents to have a function getAction(self, state) returning an action to the model.

3.2 USE CASE DIAGRAM

Use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system.

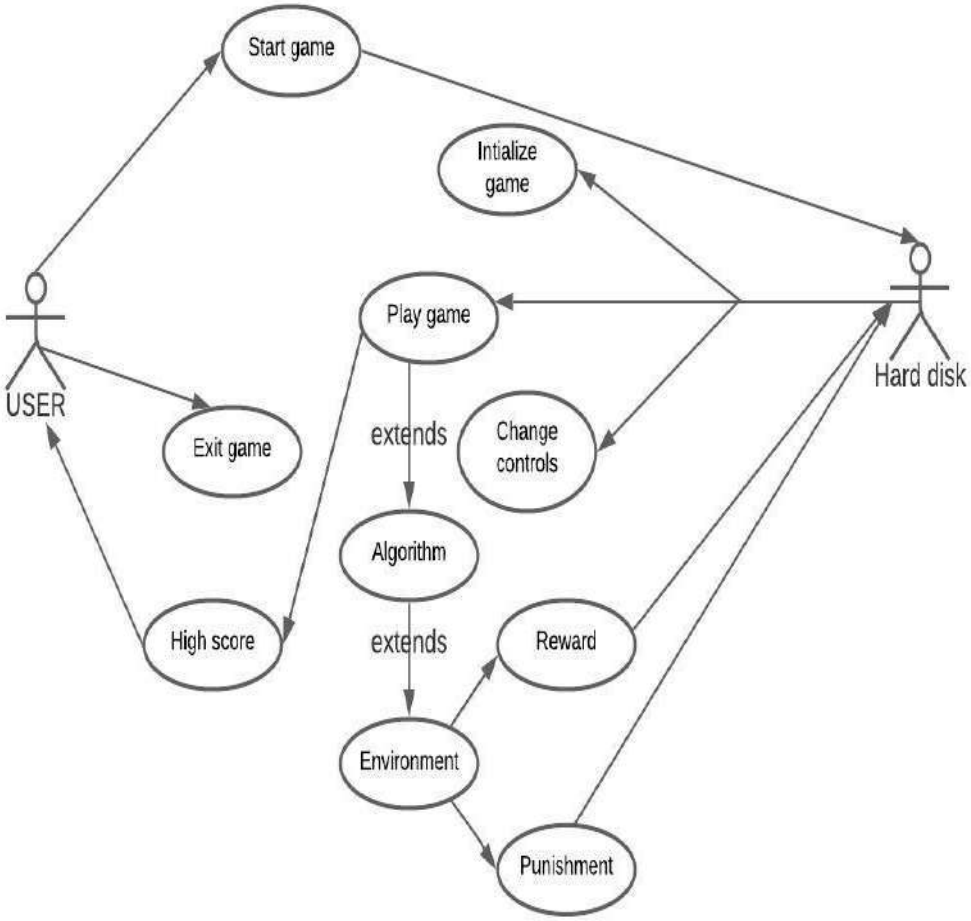

Figure 3.2 Use case diagram for Automating pacman with deep q-learning using pygame

3.3 SEQUENCE DIAGRAM

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

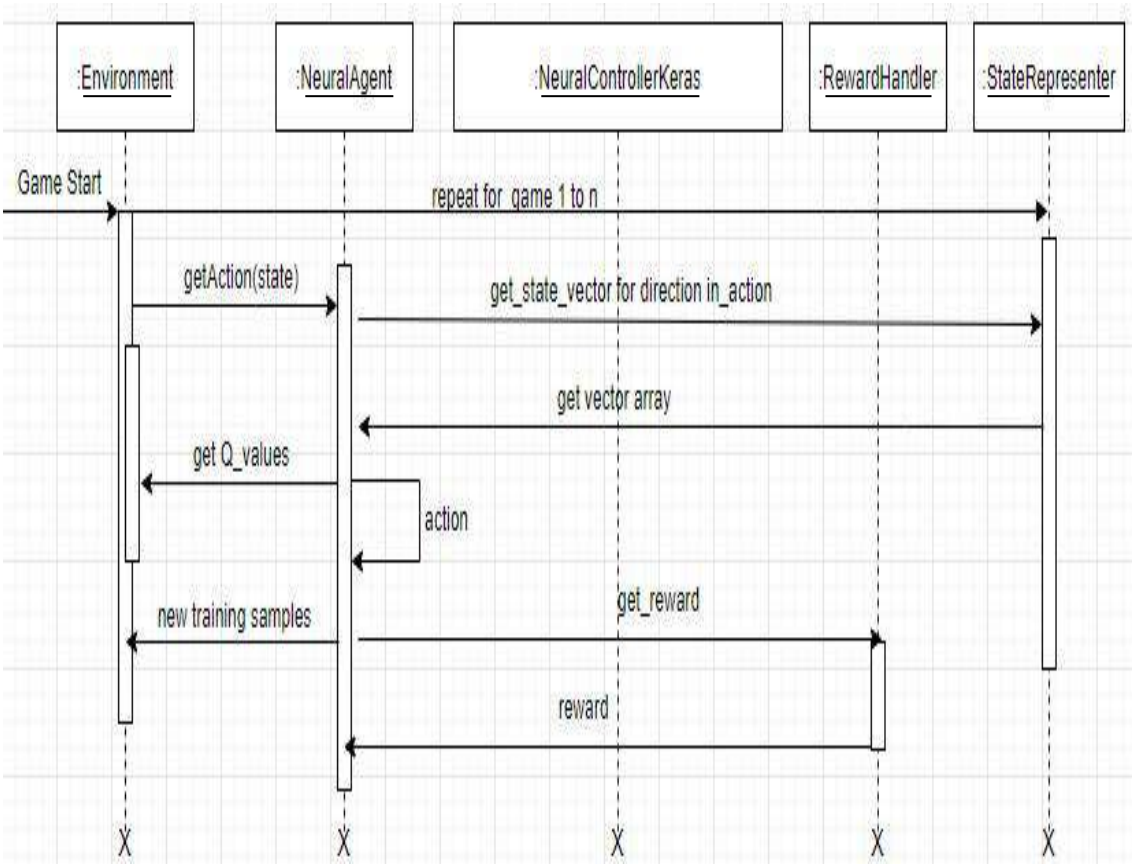

Figure 3.3 Sequence diagram for Automating pacman with deep q-learning using pygame

3.4 CLASS DIAGRAM

Class diagrams are the main building block in object-oriented modeling. In the example, a class called “loan account” is depicted. Classes in class diagrams are represented by boxes that are partitioned into three: The top partition contains the name of the class. The middle part contains the class's attributes.

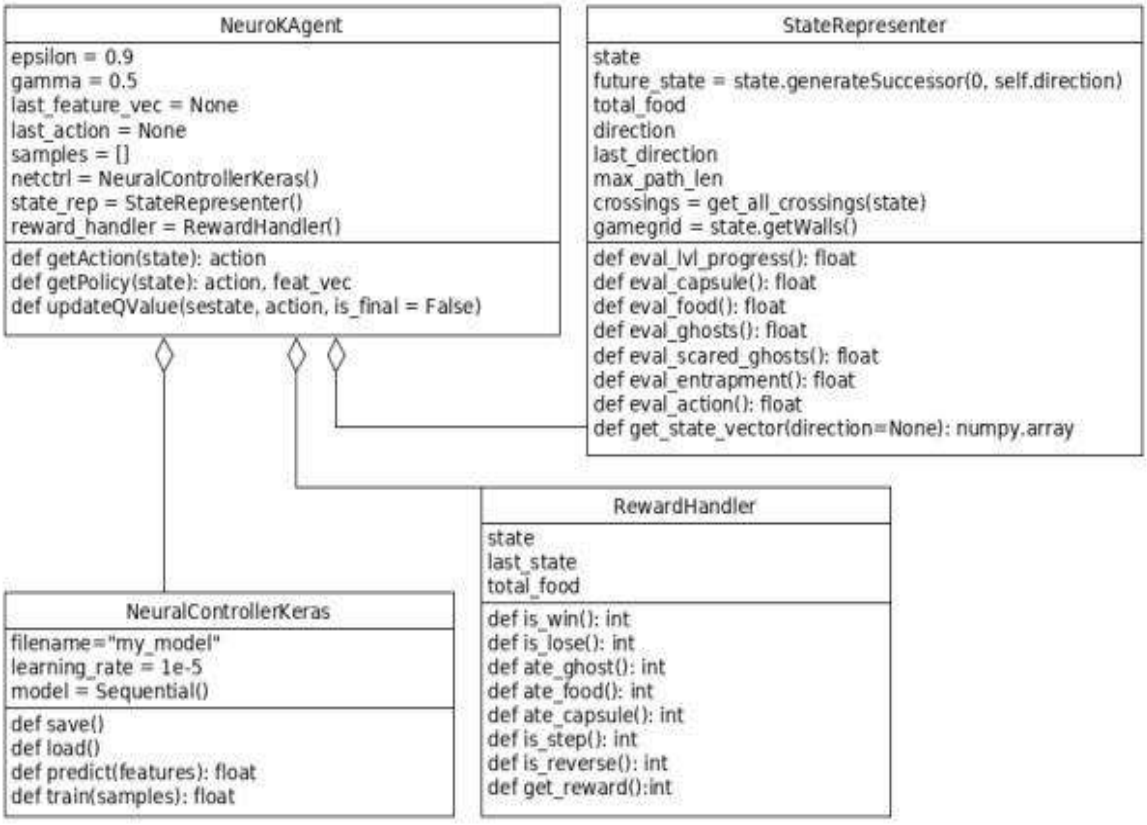

Figure 3.4 Class diagram for Automating pacman with deep q-learning using pygame

3.5 ACTIVITY DIAGRAM

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.

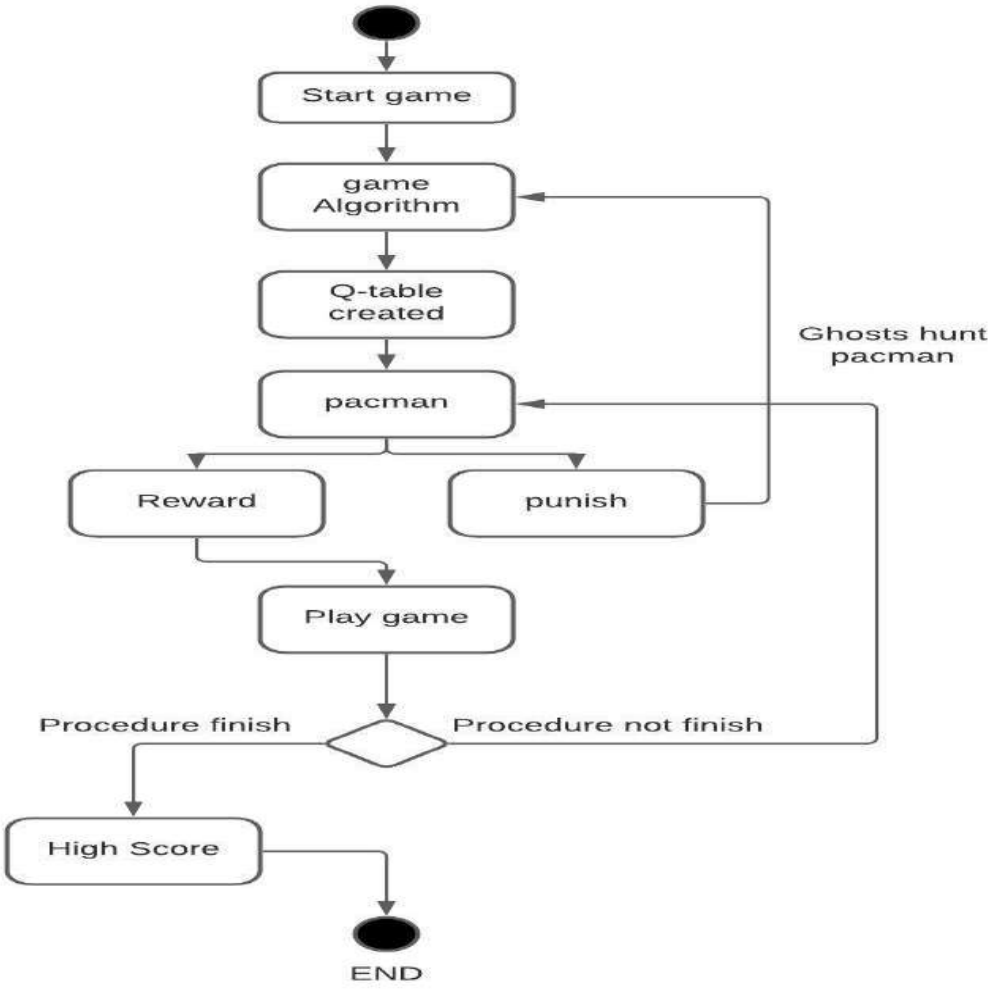

Figure 3.4 Activity diagram for Automating pacman with deep q-learning using pygame

4.IMPLEMENTATION

4.IMPLEMENTATION

4.1 SAMPLE CODE

```
import numpy as np
import random
import util
import time
import sys

# Pacman game
from pacman import Directions
from game import Agent
import game
# Replay memory
from collections import deque
# Neural nets
import tensorflow as tf
from DQN import *
params = {

    'load_file':'./saves/model-checkpoint_600497_1394',
    'save_file':'checkpoint',
    'save_interval' : 10000,

    'train_start': 5000,    # Episodes before training starts
    'batch_size': 32,       # Replay memory batch size
    'mem_size': 100000,     # Replay memory size

    'discount': 0.95,       # Discount rate (gamma value)
    'lr': .0002,            # Learning reaGHte
    # 'rms_decay': 0.99,      # RMS Prop decay (switched to adam)
    # 'rms_eps': 1e-6,        # RMS Prop epsilon (switched to adam)

    'eps': 1.0,             # Epsilon start value
    'eps_final': 0.1,       # Epsilon end value
    'eps_step': 10000       # Epsilon steps between start and end (linear)
}
```

```
class PacmanDQN(game.Agent):
    def __init__(self, args):

        print("Initialise DQN Agent")
        self.params = params
        self.params['width'] = args['width']
        self.params['height'] = args['height']
        self.params['num_training'] = args['numTraining']
       gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.1)
        self.sess = tf.Session(config = tf.ConfigProto(gpu_options = gpu_options))
        self.qnet = DQN(self.params)
        self.general_record_time = time.strftime("%a_%d_%b_%Y_%H_%M_%S",
time.localtime())
        # Q and cost
        self.Q_global = []
        self.cost_disp = 0
        # Stats
        self.cnt = self.qnet.sess.run(self.qnet.global_step)
        self.local_cnt = 0
        self.numeps = 0
        self.last_score = 0
        self.s = time.time()
        self.last_reward = 0.
        self.replay_mem = deque()
        self.last_scores = deque()
def get_value(self, direction):
        if direction == Directions.NORTH:
            return 0.
        elif direction == Directions.EAST:
            return 1.
        elif direction == Directions.SOUTH:
            return 2.
        else:
            return 3.
    def get_direction(self, value):
        if value == 0.:
            return Directions.NORTH
        elif value == 1.:
            return Directions.EAST
        elif value == 2.:
            return Directions.SOUTH
        else:
            return Directions.WEST
```

```
def observation_step(self, state):
        if self.last_action is not None:
            # Process current experience state
            self.last_state = np.copy(self.current_state)
            self.current_state = self.getStateMatrices(state)
            # Process current experience reward
            self.current_score = state.getScore()
            reward = self.current_score - self.last_score
            self.last_score = self.current_score
            if reward > 20:
                self.last_reward = 50.    # Eat ghost   (Yum! Yum!)
            elif reward > 0:
                self.last_reward = 10.    # Eat food    (Yum!)
            elif reward < -10:
                self.last_reward = -500.  # Get eaten   (Ouch!) -500
                self.won = False
            elif reward < 0:
                self.last_reward = -1.    # Punish time (Pff..)

            if(self.terminal and self.won):
                self.last_reward = 100.
            self.ep_rew += self.last_reward
            # Store last experience into memory
            experience = (self.last_state, float(self.last_reward), self.last_action,
self.current_state, self.terminal)
            self.replay_mem.append(experience)
            if len(self.replay_mem) > self.params['mem_size']:
                self.replay_mem.popleft()
            # Save model
            if(params['save_file']):
                if self.local_cnt > self.params['train_start'] and self.local_cnt %
self.params['save_interval'] == 0:
                    self.qnet.save_ckpt('saves/model-' + params['save_file'] + "_" +
str(self.cnt) + '_' + str(self.numeps))
                    print('Model saved')

            # Train
            self.train()
        # Next
        self.local_cnt += 1
        self.frame += 1
        self.params['eps'] = max(self.params['eps_final'],
                                 1.00 - float(self.cnt)/ float(self.params['eps_step']))
```

```
if self.params['load_file'] is not None:

self.global_step=tf.Variable(int(self.params['load_file'].split('_')[1]),name='global_ste
p', trainable=False)
        else:
            self.global_step = tf.Variable(0, name='global_step', trainable=False)

        # self.optim =
tf.train.RMSPropOptimizer(self.params['lr'],self.params['rms_decay'],0.0,self.params['
rms_eps']).minimize(self.cost,global_step=self.global_step)
        self.optim = tf.train.AdamOptimizer(self.params['lr']).minimize(self.cost,
global_step=self.global_step)
        self.saver = tf.train.Saver(max_to_keep=0)

        self.sess.run(tf.global_variables_initializer())

        if self.params['load_file'] is not None:
            print('Loading checkpoint...')
            self.saver.restore(self.sess,self.params['load_file'])



    def train(self,bat_s,bat_a,bat_t,bat_n,bat_r):
        feed_dict={self.x: bat_n, self.q_t: np.zeros(bat_n.shape[0]), self.actions: bat_a,
self.terminals:bat_t, self.rewards: bat_r}
        q_t = self.sess.run(self.y,feed_dict=feed_dict)
        q_t = np.amax(q_t, axis=1)
        feed_dict={self.x: bat_s, self.q_t: q_t, self.actions: bat_a, self.terminals:bat_t,
self.rewards: bat_r}
        _,cnt,cost = self.sess.run([self.optim,
self.global_step,self.cost],feed_dict=feed_dict)
        return cnt, cost

    def save_ckpt(self,filename):
        self.saver.save(self.sess, filename)
def registerInitialState(self, state): # inspects the starting state

        # Reset reward
        self.last_score = 0
        self.current_score = 0
        self.last_reward = 0.
        self.ep_rew = 0

        # Reset state
        self.last_state = None
```

```
        self.current_state = self.getStateMatrices(state)

        # Reset actions
        self.last_action = None

        # Reset vars
        self.terminal = None
        self.won = True
        self.Q_global = []
        self.delay = 0

        # Next
        self.frame = 0
        self.numeps += 1

    def getAction(self, state):
        move = self.getMove(state)

        # Stop moving when not legal
        legal = state.getLegalActions(0)
        if move not in legal:
            move = Directions.STOP

        return move
```

5.RESULTS

5.RESULTS

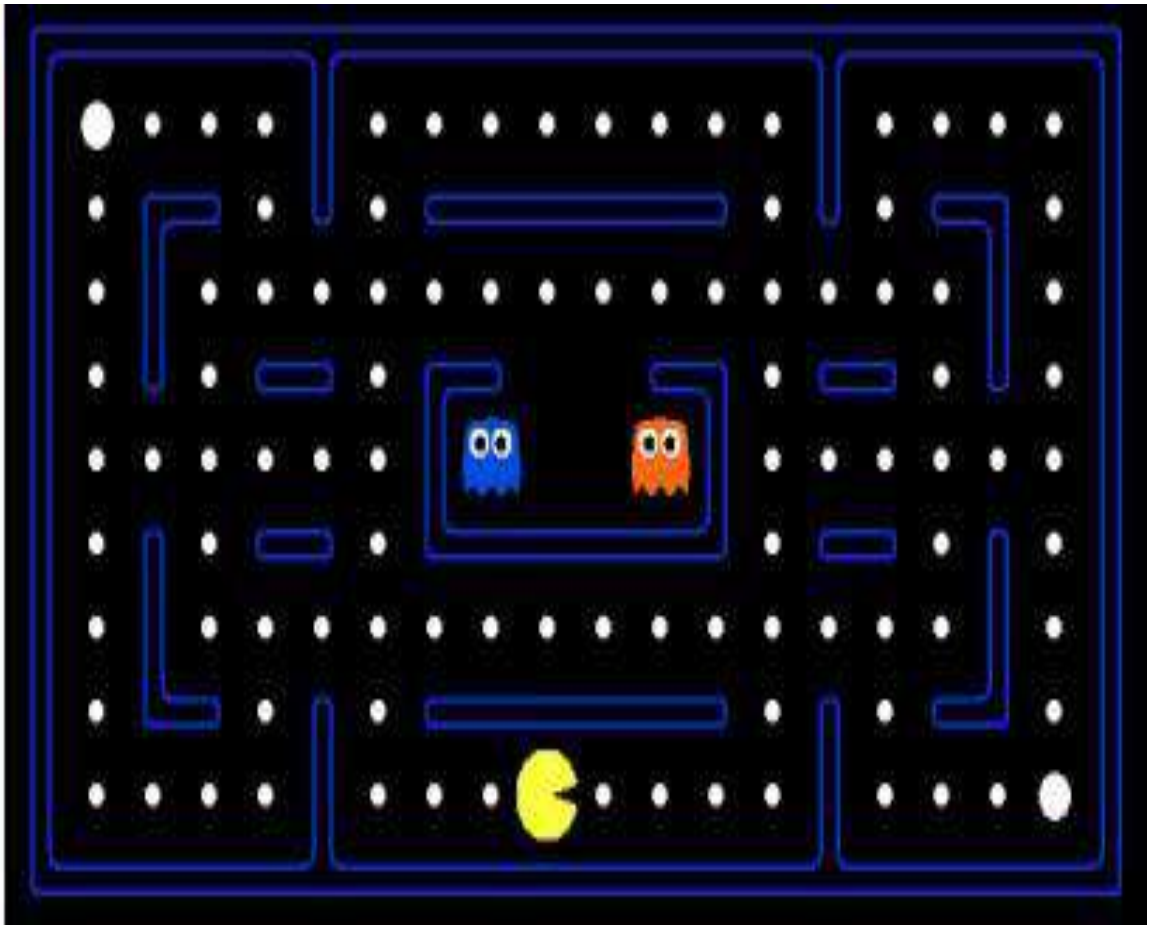

Figure 5.1: Medium Map

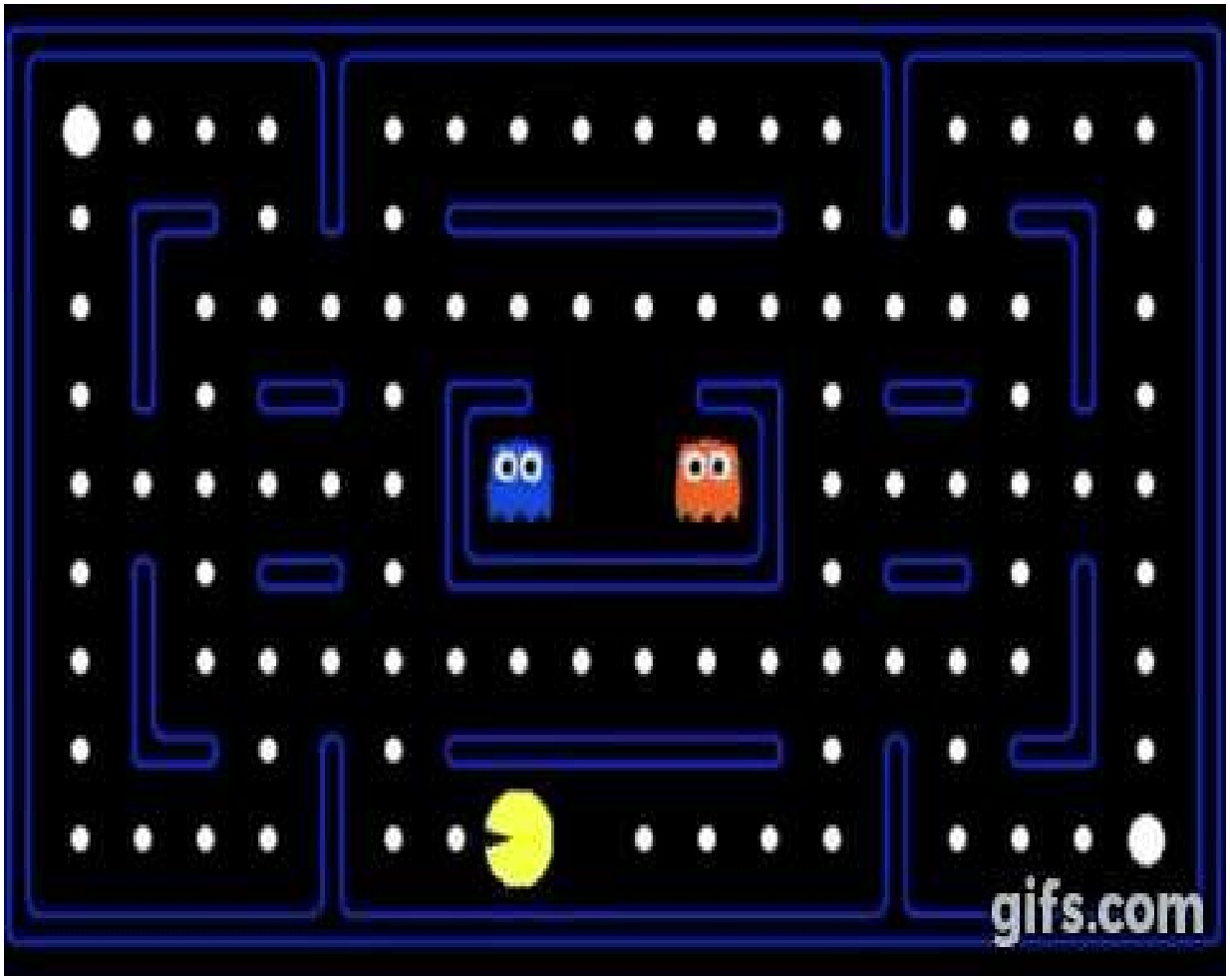

Figure 5.2: Automatic Game Play of Pacman

6.TESTING

6.TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discovery conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application , and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is cantered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

7.CONCLUSION

7.CONCLUSION

In this instance these components were linear function approximation and deep learning modules. Though the second approach was pursued mostly, since the focus laid on deep learning as an alternative for classic regression algorithms. First linear function approximation agents stagnated pretty fast in learning and performed poorly, therefore got dismissed in the early stages. This indicates that linear function approximation wasn't refined enough for this particular task. I used a simple feed forward network, mostly utilized for regression and classification tasks, to estimate the beneficially of possible feature combinations for estimating which action given the current state might be the best. The first attempted artificial network agent, trained on data sets of feature combinations mapped to received rewards, averagely outscored the best linear function approximation agent's score by triple the points.

8. BIBILOGRAPHY

8.BIBILOGRAPHY

8.1 REFERENCES

1. https://towardsdatascience.com/automating-pac-man-with-deep-q-learning-an-implementation-in-tensorflow-ca08e9891d9c

2. https://en.wikipedia.org/wiki/Pygame

3. https://pythonrepo.com/tag/unity-game

4. https://www.reddit.com/r/learnmachinelearning/comments/eehhml/auto-mating_pacman_with_deep_qlearning.

8.2 GITHUB LINK

https://github.com/nandyalasaikumar/PACMAN.git