**การฝึกพฤติกรรมของตัวละครที่ผู้เล่นไม่ได้ควบคุมในเกมโดยวิธีการเรียนแบบเสริมกำลัง**

**(NON-PLAYER CHARACTER BEHAVIOR TRAINING IN GAME USING REINFORCEMENT LEARNING)**

**โดย**

**สุรเชษฐ์ ใหญ่ธรรมสาร**

**อัครพล อัครสูริย์**

**อาจารย์ที่ปรึกษา**

**ดร. สามารถ หมุดและ**

**ดร. สุพัณณดา โชติพันธ์**

**ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต**

**สาขาวิชาเทคโนโลยีสารสนเทศ**

**คณะเทคโนโลยีสารสนเทศ**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ภาคเรียนที่ 1 ปีการศึกษา 2562**

**NON-PLAYER CHARACTER BEHAVIOR TRAINING IN GAME USING REINFORCEMENT LEARNING**

**SURACHET YAITAMMASAN**

**AKARAPON AKARASURI**

**A PROJECT SUBMITTED IN PARTIAL FULFILLMENT**

**OF THE REQUIREMENT FOR THE DEGREE OF**

**BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY**

**FACULTY OF INFORMATION TECNOLOGY**

**KING MONGKUT’S INSTITUTE OF TECHNOLOGY LADKRABANG**

**1/2019**

**COPYRIGHT 2019**

**FACULTY OF INFORMATION TECHNOLOGY**

**KING MONGKUT’S INSTITUTE OF TECHNOLOGY LADKRABANG**

**ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2562**

**คณะเทคโนโลยีสารสนเทศ**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**เรื่อง การฝึกพฤติกรรมของตัวละครที่ผู้เล่นไม่ได้ควบคุมในเกมโดยวิธีการ เรียนแบบเสริมกำลัง**

**NON-PLAYER CHARACTER BEHAVIOR TRAINING IN GAME USING REINFORCEMENT LEARNING**

**ผู้จัดทำ**

1. **นาย สุรเชษฐ์ ใหญ่ธรรมสาร รหัสนักศึกษา 59070180**
2. **นาย** **อัครพล อัครสูริย์ รหัสนักศึกษา 59070189**

**.........................................อาจารย์ที่ปรึกษา**

(.......................................)

**.........................................อาจารย์ที่ปรึกษาร่วม**

(.......................................)

**ใบรับรองโครงงาน (PROJECT)**

**เรื่อง**

**การฝึกพฤติกรรมของตัวละครที่ผู้เล่นไม่ได้ควบคุมในเกมโดยวิธีการเรียนแบบเสริมกำลัง**

**NON-PLAYER CHARACTER BEHAVIOR TRAINING IN GAME USING REINFORCEMENT LEARNING**

**นาย สุรเชษฐ์ ใหญ่ธรรมสาร รหัสนักศึกษา 59070180**

**นาย อัครพล อัครสูริย์ รหัสนักศึกษา 59070189**

ขอรับรองว่ารายงานฉบับนี้ ข้าพเจ้าไม่ได้คัดลอกมาจากที่ใด

รายงานฉบับนี้ได้รับการตรวจสอบและอนุมัติให้เป็นส่วนหนึ่งของ

การศึกษาวิชาโครงงาน หลักสูตรวิทยาศาสตรบัณฑิต (เทคโนโลยีสารสนเทศ)

ภาคเรียนที่ 1 ปีการศึกษา 2562

……………………………….

(นาย สุรเชษฐ์ ใหญ่ธรรมสาร)

……………………………….

(นาย อัครพล อัครสูริย์)

**หัวข้อวิทยานิพนธ์** การฝึกพฤติกรรมของตัวละครที่ผู้เล่นไม่ได้ควบคุมในเกมโดยวิธีการ เรียนรู้แบบเสริมกำลัง

**นักศึกษา** นาย สุรเชษฐ์ ใหญ่ธรรมสาร **รหัสนักศึกษา** 59070180

นาย อัครพล อัครสูริย์ **รหัสนักศึกษา** 59070189

**ปริญญา** วิทยาศาสตรบัณฑิต

**สาขาวิชา** เทคโนโลยีสารสนเทศ

**พ.ศ.** 2562

**อาจารย์ที่ปรึกษา** ดร. สามารถ หมุดและ

**อาจารย์ที่ปรึกษาร่วม** ดร. สุพัณณดา โชติพันธ์

**บทคัดย่อ**

ในปัจจุบันวีดิโอเกมเป็นอุตสาหกรรมสื่อบันเทิงรูปแบบหนึ่งที่มีขนาดใหญ่ขึ้นอย่างต่อเนื่องในช่วงเวลาที่ผ่านมา ทำให้มีผู้คนสนใจที่จะพัฒนาเพิ่มขึ้น ทำให้เกิดการนำ Machine learning มาประยุกต์ใช้ร่วมกับวีดิโอเกมมากยิ่งขึ้น โดยพี้นฐานความยากง่ายของตัววีดิโอเกมส่วนใหญ่จะมาจากผู้พัฒนาสร้างสคริปต์จะไม่มีความยากง่ายเกินกว่าที่ผู้เล่นทำการเลือก

จนกระทั่งในปี พ.ศ.2556 (ค.ศ.2013) กลุ่ม DeepMind ได้สร้างโมเดลที่ชื่อว่า Deep-Q Learning เพื่อนำมาทดสอบกับเกมของเครื่อง Atari 2600 ซึ่งได้ผลดี ทำให้เป็นจุดเริ่มต้นของการนำการเรียนรู้แบบเสริมกำลังมาใช้งานร่วมกัน ต่อมาได้มีการพัฒนาต่อยอดมาเป็น AlphaStar เป็น AI ของเกม Starcraft 2 โดยนำมาทดสอบกับนักแข่งมืออาชีพและได้ผลลัพธ์ที่น่าพอใจ

ผู้จัดทำจึงมีต้องการที่จะนำเสนอการเรียนรู้เบื้องต้นของการเรียนรู้แบบเสริมกำลังโดยใช้วีดิโอเกมที่มีความละเอียดของภาพต่ำ และมีความซับซ้อนของการเล่นที่น้อย และนำวิธีการเรียนรู้พื้นฐานของการเรียนรู้แบบเสริมกำลังมาทำการสอนให้คอมพิวเตอร์เรียนรู้และสามารถเล่นได้เอง ผู้จัดต้องการที่จะทราบว่าวิธีการเรียนรู้แบบเสริมกำลังแบบใดเหมาะกับเกมที่นำมาใช้เป็นสภาพแวดล้อมในการเล่น

**Project Title** NON-PLAYER CHARACTER BEHAVIOR TRAINING IN GAME USING REINFORCEMENT LEARNING

**Student** Surachet Yaitammasan **Student ID** 59070180

Akarapon Akarasuri **Student ID** 59070189

**Degree** Bachelor of Science

**Program** Information Technology

**Academic Year** 2019

**Project Advisor** Dr. Samart Moodleah

**Project Advisor (Co)** Dr. Supannada Chotipant

**ABSTRACT**

Video game is a part of Entertainment Industries are getting bigger and bigger nowdays. Developers are interest to implement a Machine Learning to video games, Basiclly a difficulty in video games are being scripted. Difficulty can’t take more difficult than a player choice.

In 2013, Deepmind create a model called “Deep Q Network” and tested with Atari 2600 games and a result are effectively. Afterward, Deepmind created a “AlphaStar” is a Artificial Intelligence for Starcraft 2, A Real-Time Strategy game. AlphaStar is getting evaluated by played with Competitive Starcraft 2 players. Results are excellent.

We want to represent a basic of Reinforcement Learning by using a low resolution and less complicate video game to trains a computer via Reinforcement Learning method to find which methods are suite an environment

**กิตติกรรมประกาศ**

ปริญญานิพธ์นี้สำเร็จลุล่วงได้ด้วยความกรุณาจากด็อกเตอร์ สามารถ หมุดและ และด็อกเตอร์ สุพัณณดา โชติพันธ์ อาจารย์ที่ปรึกษาโครงงานที่ได้ให้คำแนะนำ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่างๆ มาโดยตลอด จนโครงงานเล่มนี้เสร็จสมบูรณ์ ผู้ศึกษาจึงขอกราบขอบพระคุณเป็นอย่างสูง  
 ขอขอบพระคุณคณาจารย์คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกๆ ท่านให้ความรู้กับผู้จัดทำ  
 ขอขอบคุณเพื่อนๆ ที่ให้คำปรึกษาการเรื่องการทำงาน และกำลังใจที่ดีเสมอมา  
 ขอขอบคุณ David Silver ผู้เชี่ยวชาญทางด้านการเรียนรู้แบบเสริมกำลัง ที่มอบแนวคิด ทฤษฏี และให้ความรู้เกี่ยวกับ การเรียนรู้แบบเสริมกำลัง  
 และความดีอันเกิดจากการศึกษาค้นคว้าครั้งนี้ ผู้เขียนขอมอบแด่ผู้ที่มีความประสงค์ที่จะเรียนรู้เกี่ยวกับการเรียนรูแบบเสริมกำลัง ผู้เขียนมีความซาบซึ้งในความกรุณาอันยิ่งใหญ่จากทุกท่านที่ได้กล่าวนามมาและขอกราบขอบพระคุณมา ณ โอกาศนี้

นาย สุรเชษฐ์ ใหญ่ธรรมสาร

นาย อัครพล อัครสูริย์

สารบัญ

[สารบัญรูป 5](#_Toc21347378)

[บทที่ 1 บทนำ 6](#_Toc21347379)

[1.1 ที่มาและความสำคัญ 6](#_Toc21347380)

[1.2 วัตถุประสงค์ 6](#_Toc21347381)

[1.3 ขอบเขตของโครงงาน 6](#_Toc21347382)

[1.4 ประโยชน์ที่คาดว่าจะได้รับ 6](#_Toc21347383)

[บทที่ 2 ทฤษฏีและงานวิจัยที่เกี่ยวข้อง 7](#_Toc21347384)

[2.1 ทฤษฏี 7](#_Toc21347385)

[2.2 งานวิจัยที่เกี่ยวข้อง 12](#_Toc21347386)

[บทที่ 3 วิธีการดำเนินงาน 15](#_Toc21347387)

[3.1 โปรแกรมหรือซอฟต์แวร์ที่ใช้ในการพัฒนา 15](#_Toc21347388)

[3.2 ขั้นตอนการดำเนินงาน 15](#_Toc21347389)

[บรรณาณุกรม 17](#_Toc21347390)

# **สารบัญรูป**

หน้า

รูปที่

* 1. โครงสร้างการทำงานของการเรียนรู้แบบเสริมกำลัง 8
  2. โครงสร้างของห่วงโซ่มาร์คอฟ 9

2.3 โครงสร้างของโครงข่ายประสาทแบบคอนโวลูชัน 16

2.4 รูปภาพโครงสร้างของการเรียนรู้แบบเสริมกำลังเชิงลึก 14

3.1 ภาพของเกม Kaboom จากเครื่อง Atari2600 19

3.2 โครงสร้างของ ActionWrapper และ Discretizer.py 20

3.3 โครงสร้างของสภาพแวดล้อมและ Gym\_Wrapper.py 21

**สารบัญตาราง**

หน้า

ตารางที่

3.1 ลักษณะโครงสร้างของสภาพแวดล้อมของเกม Kaboom 19

3.2 โครงสร้างของสภาพแวดล้อมหลังจากทำการแยกการกระทำที่ต้องการ 20

3.3 โครงสร้างของสภาพแวดล้อมหลังจากการปรับ Observation สำหรับโครงข่ายคอนโวลูชัน 21

# **บทที่ 1 บทนำ**

บทนี้จะกล่าวถึงที่มาและความสำคัญ รวมไปถึงวัตถุประสงค์และประโยชน์ที่คาดว่าจะได้รับจากวิจัยครั้งนี้เพื่อทราบถึงจุดมุ่งหมายที่แท้จริงของผู้วิจัยโดยที่มีรายละเอียดของการดำเนินงานและขอบเขตรวมไปถึงอุปกรณ์ที่ใช้เพื่อเป็นประโยชน์แก่ผู้สนใจในการศึกษางานวิจัยนี้โดยมี รายละเอียดดัง ต่อไปนี้

## **1.1 ที่มาและความสำคัญ**

การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) เป็นหนึ่งในแขนงของ Machine Learning ที่ถูกนำมาใช้กับอุตสาหกรรมวิดีโอเกมมากขึ้น เช่น AlphaGo เป็นต้น ผู้จัดทำมีความประสงค์ในการพัฒนาองค์ความรู้ที่เกี่ยวข้อง เพื่อพัฒนา Algorithm ที่สามารถฝึกตัวละครในเกมที่กำหนดเพื่อเพิ่มขีดความสามารถในการเล่นเกมให้เทียบเคียงกับมนุษย์ ซึ่งองค์ความรู้ที่กำหนดได้สามารถนำไปประยุกต์ใช้ได้ในหลากหลายสาขาในอนาคต เช่น Robot Control เป็นต้น

ความนิยมของการเรียนรู้แบบเสริมกำลังมาจากทาง OpenAI ได้ทำการเปิดตัว

OpenAI Five ซึ่งเป็นปัญญาประดิษฐ์ที่สร้างมาสำหรับการเล่นเกม DOTA2 ซึ่งใช้โครงสร้างและหลักการของการเรียนรู้แบบเสริมกำลังและการเล่นของตัวปัญญาประดิษฐ์ภายใต้การคำนวณของซีพียูมากกว่าหนึ่งแสนตัว และตัวเกมมีความซับซ้อนที่สูงถึงแม้ว่าจะสามารถเล่นคนเดียวก็ตาม แต่หัวใจสำคัญคือความซับซ้อนที่ต้องใช้ความเข้าใจและประสบการณ์ในการเล่น เป้าหมายของเกม ด้วยพื้นฐานของเกมเป็นการเล่นแบบทีม ทำให้มีความซับซ้อนที่มากกว่าเดิม

เป้าหมายของเกม DOTA2 คือ การจัดการทีมอีกฝ่าย ทำลายสิ่งปลูกสร้างที่อยู่ในฐานของทีมอีกฝ่ายแต่จนกว่าจะไปถึงเป้าหมายนั้นจะมีรายละเอียดเล็กน้อยหรือเป้าหมายย่อยที่ทำให้สามารถสำเร็จเป้าหมายของเกมได้

ทางผู้จัดทำสนใจที่ทำการเรียนรู้การเรียนรู้แบบเสริมกำลังด้วยมีสภาพแวดล้อมให้กับปัญญาประดิษฐ์ด้วยวิดีโอเกม แต่ด้วยข้อจำกัดของอุปกรณ์ ผู้จัดทำนำเกมที่มีความซับซ้อนน้อยลงโดยเป็นเกมที่อยู่ในยุคเริ่มต้นของอุตสากรรมวิดีโอเกมที่มีความละเอียดและความซับซ้อนที่น้อยลง

เพื่อนำมาศึกษาหลักการและอัลกอริทึมที่เกี่ยวกับการเรียนรู้แบบเสริมกำลัง

## **1.2 วัตถุประสงค์**

1. เพื่อพัฒนาระบบปัญญาประดิษฐ์กับวีดิโอเกมด้วยวิธีการ Reinforcement Learning

2. เพื่อพัฒนาองค์ความรู้ด้านการ Reinforcement Learning

3. ศึกษาหลักการและโครงสร้างของการเรียนรู้แบบเสริมกำลังผ่านวิดีโอเกม

## **1.3 ขอบเขตของโครงงาน**

1. ออกแบบวิธีการด้วยวิธีการเรียนรู้แบบเสริมกำลังสำหรับพัฒนาพฤติกรรมของ NPC

2. เปรียบเทียบผลของ NPC ที่ใช้อัลกอริทึมที่ต่างกันของวิธีการเรียนรู้แบบเสริมกำลัง

3. วิเคราะห์ผลของเกมที่นำ NPC ที่ผ่านการพัฒนาโดยอัลกอริทึมต่างๆ ของการเรียนรู้แบบเสริมกำลัง

## **1.4 ประโยชน์ที่คาดว่าจะได้รับ**

อุตสาหกรรมเกมในปัจจุบัน ความสามารถของเกมและ AI ยังคงมาจากการเขียนสคริปต์จากผู้พัฒนาเกม ผู้จัดทำโครงการจึงต้องการสร้างพื้นฐานของ AI ที่ทำการเรียนรู้โดยการลองผิดลองถูก ซึ่งเป็นรากฐานของ Reinforcement Learning เพื่อเป็นรากฐานในการนำ Reinforcement Learning ไปใช้เพื่อเพิ่มความท้าทายของตัวเกมต่อไป

# **บทที่ 2 ทฤษฏีและงานวิจัยที่เกี่ยวข้อง**

## **2.1 ทฤษฏี**

### **2.1.1 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)**

การเรียนรู้แบบเสริมกำลังเป็นส่วนหนึ่งของการเรียนรู้ด้วยเครื่อง (Machine Learning) ที่ทำการโดยนำปัญญาประดิษฐ์นำมาอยู่ภายใต้สภาพแวดล้อมหนึ่งที่ปัญญาประดิษฐ์ทำการตัดสินใจในการกระทำหนึ่ง ที่มาจากการสุ่มหรือ เลือกการกระทำจากข้อมูลที่อยู่ภายใต้สิ่งแวดล้อมที่กำหนด ซึ่งปัญญาประดิษฐ์ของการเรียนรู้แบบเสริมกำลังมีเป้าหมายคือ เลือกการกระทำที่ทำให้รับรางวัลที่ดีที่สุดในการแก้ปัญหาหนึ่ง ผ่านการลองผิดลองถูกของตัวปัญญาประดิษฐ์

การเรียนรู้แบบเสริมกำลังส่วนใหญ่จะถูกใช้ในอุตสาหกรรมหุ่นยนต์ และอุตสาหกรรมวิดีโอเกม เช่น ใช้การเรียนรู้แบบเสริมกำลังสร้างปัญญาประดิษฐ์ในการเล่นเกมคอมพิวเตอร์ Starcraft 2 หรือ สร้างปัญญาประดิษฐ์ควบคุมการทำงานของแขนกล

ซึ่งองค์ประกอบของการเรียนรู้แบบเสริมกำลังมีทั้งหมด 5 ส่วน

#### **2.1.1.1 เอเจนต์ (Agent)**

ปัญญาประดิษฐ์ที่อยู่ภายใต้สภาพแวดล้อมและการทำงานของการเรียนรู้แบบเสริม กำลัง ซึ่งภายในเอเจนต์หนึ่งตัวจะมีส่วนประกอบภายใน 1 ประเภทหรือมากกว่า ซึ่งมีทั้งหมด 3 องค์ประกอบดังนี้

##### **2.1.1.1.1 Policy**

เป็นวิธีการที่จะให้ไปถึงเป้าหมายที่ต้องการเมื่ออยู่ในสถานะที่ต่างกันออกไป

##### **2.1.1.1.2 Value Function**

เป็นค่าที่ใช้วัดผลจากการกระทำในสถานะต่าง ๆ เพื่อวัดผลว่าถ้าทำการกระทำหนึ่ง ณ สถานะปัจจุบันให้ผลดีต่อรางวัลในอนาคตอย่างไร

##### **2.1.1.1.3 Model**

โมเดลเป็นการทำนายว่าในสภาพแวดล้อมจะเกิดอะไรขึ้นต่อไปทั้งสถานะและรางวัลที่จะได้จากการกระทำ

#### **2.1.1.2** **สภาพแวดล้อม (Environment)**

เป็นพื้นที่ทำการนำปัญญาประดิษฐ์ทำภารกิจในสภาพแวดล้อมที่กำหนด

#### **2.1.1.3 สถานะ (State)**

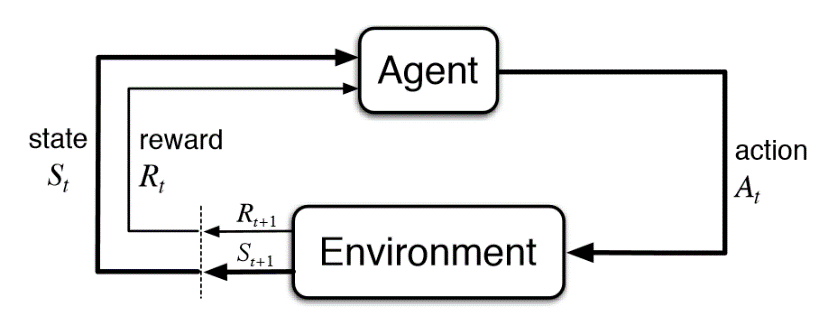
เป็นสถานะของสภาพแวดล้อมในช่วงเวลาต่างๆที่ปัญญาประดิษฐ์สามารถรับรู้เพื่อตัดสินใจเลือกการกระทำในแต่ละช่วงเวลา

#### **2.1.1.4 การกระทำ (Action)**

เอเจนต์จะเลือกการกระทำที่ทำการตัดสินใจจากสถานะก่อนหน้าเข้าไปยังสภาพแวดล้อมเพื่อแสดงสถานะถัดไปและรางวัลที่ได้จากการเลือกการกระทำ

#### **2.1.1.5 รางวัล (Reward)**

เป็นรางวัลที่ได้จากการกระทำเพื่อแสดงว่าการกระทำที่เลือกไปดีต่อกับสภาพแวดล้อมและเป้าหมายของสภาพแวดล้อมได้ดีเพียงใด



**รูป 2.1** โครงสร้างการทำงานของการเรียนรู้แบบเสริมกำลัง

ซึ่งการแก้ปัญหาที่ให้การเรียนรู้แบบเสริมกำลังส่วนใหญ่มักจะแก้ไขด้วยหลักการที่ชื่อว่า กระบวนการตัดสินใจของมาร์คอฟ (Markov Decision Process) ซึ่งมีคุณสมบัติดังนี้

สถานะใดๆ สถานะหนึ่งจะเป็นสถานะมาร์คอฟได้ก็ต่อเมื่อ สถานะปัจจุบันนั้นส่งผลถึงสถานะในอนาคต เท่ากับสถานะปัจจุบันและสถานะก่อนหน้าส่งผลกับสถานะในอนาคต ซึ่งทำให้ไม่ต้องสนใจสถานะในอดีตอีกต่อไป

(1)

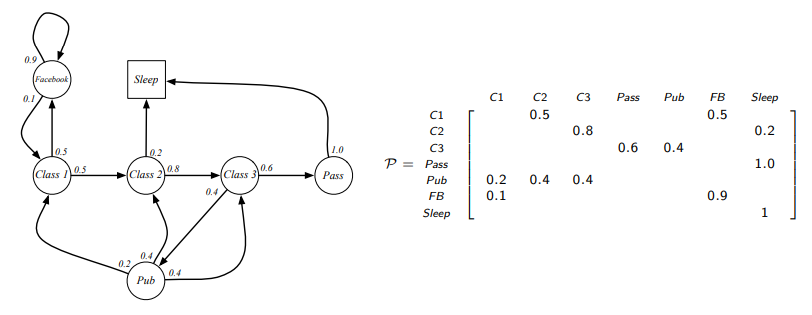
State Transition Matrix เป็นเมทริกซ์ของความน่าจะเป็นของสถานะที่อยู่ไปยังสถานะต่อไปซึ่งแต่ละจุดจะมีความน่าจะเป็นดังสมการ

(2)

ซึ่งแต่ละจุดสามารถรวมเป็นเมทริกซ์ได้ดังนี้

(3)

ซึ่งในแต่ละจุดของเมทริกซ์ จะเป็นความน่าจะเป็นของสถานะปัจจุบัน (แถวของเมทริกซ์) ไปยังสถานะถัดไป (คอลลัมน์ของเมทริกซ์) ซึ่งผลรวมของแต่ละแถวจะมีผลรวมของความน่าจะเป็นเท่ากับ 1 และสามารถสร้างเป็นห่วงโซ่มาร์คอฟ (Markov Chain) ได้ดังรูปที่ 2.2



**รูปที่ 2.2** โครงสร้างของห่วงโซ่มาร์คอฟ

กลุ่มของสถานะที่เดินทางตั้งแต่สถานะแรก (Initial State)ไปยังสถานะสิ้นสุด (Terminal State) ภายในห่วงโซ่มาร์คอฟนั้นเรียกว่า เอพิโซด (Episode)

(4)

โดยการวัดจากโมเดลที่สร้างมาจากการเรียนรู้แบบเสริมกำลังจะมีผลทั้งหมด 2 แบบคือ รางวัลโดยรวมในแต่ละเอพิโซด (Episodic Return) และ ค่าแวลูฟังก์ชัน (Value Function) ซึ่งรางวัล

รางวัลโดยรวมในแต่ละ Episode จะเป็นรางวัลที่คาดหวังปัจจุบันที่อยู่ในภายใต้สภาพแวดล้อมต่าง ๆที่เอเจนต์ทำงานอยู่ภายในที่อยู่ในช่วงเวลานั้น ตัวอย่าง เช่น ถ้าสภาพแวดล้อมเป็นวิดีโอเกมรางวัลของการเล่นเกมคือ ได้คะแนนเพิ่มขึ้น หรือถ้าทำการฝึกแขนกล รางวัลคือการทำงานสำเร็จในแต่ละครั้ง เป็นต้น

(5)

ซึ่งมีส่วนประกอบไปด้วย Reward Function คือค่าคาดหวังของรางวัลที่จะได้เมื่อเอเจนต์ได้ทำการกระทำที่อยู่ภายในสถานะนั้น และค่า γ เรียกว่า Discount Factor ซึ่งมีค่าระหว่าง 0 ถึง 1 โดยทำให้ค่าเป็นปัจจุบันในทุกหน่วยเวลาของรางวัลที่จะได้ในอนาคต และใช้ตัดสินใจว่า รางวัลที่ได้จากการกระทำจะเลือกรับรางวัลทันทีหรือรอรับรางวัลในภายหลัง เพื่อที่อาจจะได้รางวัลที่ดีกว่าในภายหลัง

แวลูฟังก์ชัน (Value Function) เป็นผลรวมของรางวัล ณ สถานะหนึ่งเพื่อแสดงว่าการกระทำที่เลือกมานั้นส่งผลดีหรือพาไปยังเป้าหมายได้ดีเพียงใดซึ่ง Value Function มีสองประเภทขึ้นอยู่กับการใช้งานได้แก่ State-Value Funtion และ Action-Value Function

#### State-Value Function จะเป็นผลรวมของรางวัลที่สถานะไปยังสถานะใหม่ตาม Policy π เพื่อดูว่าการเคลื่อนไปยังสถานะใหม่นั้นมีผลดีเพียงใด

(6)

Action-Value Function จะเป็นผลรวมของรางวัลที่สถานะที่เลือกการกระทำที่นำพาไปยังสถานะใหม่ตาม Policy π เพื่อดูว่าการเคลื่อนไปยังสถานะใหม่นั้นมีผลดีเพียงใด

(7)

**2.1.2 Gym และ Gym-retro คืออะไร?**

Gymเป็นไลบรารีโอเพนซอร์ซที่ทาง OpenAI สร้างขึ้นเพื่อให้ผู้ที่มีความสนใจในการพัฒนาในการเรียนรู้แบบเสริมกำลัง เพื่อพัฒนาและเปรียบเทียบอัลกอริทึมของการเรียนรู้แบบเสริมกำลังภายใต้สภาพแวดล้อมต่างๆ สภาพแวดล้อมที่ทาง Gym มีให้ก็มีด้วยกันหลายรูปแบบด้วยกัน ไม่ว่าจะเป็น เกมที่เป็นข้อความ หรือวีดิโอเกม หรือไปจนถึงการฝึกซ้อมหุ่นยนต์ โดยเป้าหมายเพื่อให้เอเจนต์(ปัญญาประดิษฐ์)สามารภทำภารกิจได้ลุล่วงตามสภาพแวดล้อมที่กำหนด

Gym-retro คือไลบรารีโอเพนซอร์ซที่ทาง OpenAI สร้างขึ้นโดยมีรากฐานเดียวกับ Gym คือ ให้ผู้ที่สนใจพัฒนาในการเรียนรู้แบบเสริมกำลัง แต่ว่ามีความแตกต่างที่สภาพแวดล้อมของ Gym-retro จะเป็นเกมคลาสสิคที่เกิดในยุคแรกของอุตสาหกรรมวีดิโอเกม(ในช่วงประมาณ ค.ศ.1970 - ค.ศ.1995) เช่น Space Invader, Super Mario Bros, Sonic The Hedgehog เป็นต้น และเกมที่อยู่ในช่วงเวลาเดียวกัน

**2.1.3 เกม Kaboom**

เกม Kaboom เป็นเกมจากเครื่อง Atari 2600 เป็นเกมในยุคเริ่มต้นของอุตสาหกรรมเกมซึ่งมีความละเอียดและความซับซ้อนของเกมที่น้อย ในเกมนี้จะมีโจรปล่อยระเบิดลงมาเพื่อให้ผู้เล่นรับระเบิดเมื่อรับสำเร็จจะได้รับคะแนน และเมื่อผู้เล่นรับไม่ได้ ผู้เล่นจะเสียพลังชีวิต 1 ชีวิต ซึ่งพลังชีวิตมีในการเล่น 1 ครั้ง มีด้วยกันทั้งหมด 3 พลังชีวิต เกมนี้มีวิธีการควบคุมคือการดันคันโยกของเครื่องเกมไปทางซ้ายและขวาเพื่อรับระเบิด และกดปุ่มเพื่อให้โจรทำการปล่อยระเบิดในชุดต่อไป

### **2.1.4 โครงข่ายประสาทแบบคอนโวลูชัน (Convolutional Neural Network)**

 โครงข่ายประสาทแบบคอนโวลูชัน เป็นโครงข่ายประเภทหนึ่งของ Deep Learning ที่ทำการจำลองการมองเห็นของมนุษย์โดยการแบ่งเป็นส่วนย่อย และนำมารวมเป็นภาพรวมเพื่อแยกประเภทหรือหมวดหมู่ มักจะใช้ในการประมวลผลภาพสำหรับการฝึกปัญญาประดิษฐ์จำแนกประเภทด้วยภาพ ทำการแบ่งภาพเป็นส่วนย่อย ๆ ในการจดจำรูปแบบในแต่ละกลุ่มของรูปหนึ่งรูป เพื่อจำแนกคุณลักษณะ เพื่อการจำแนกค่ารับเข้าในรูปแบบของรูปภาพได้

รูป 2.3 โครงสร้างของโครงข่ายประสาทแบบคอนโวลูชัน

องค์ประกอบของโครงข่ายคอนโวลูชันมีดังนี้

#### **2.1.4.1 Convolutional Layer**

เป็นชั้นที่ทำการสแกนค่ารับเข้าซึ่งเป็นรูปภาพ เพื่อแยกองค์ประกอบของรูป เช่น สี รูปทรง ขอบของภาพ

#### **2.1.4.2 Pooling Layer**

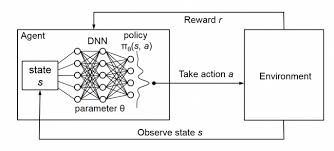
เป็นการลดขนาดของข้อมูลมีขนาดเล็กลงโดยที่รายละเอียดยังคงเดิมซึ่งทั้ง Convolutional Layer และ Pooling Layer จะทำงานคู่กันซึ่งจะทำงานหลายๆครั้งเพื่อจำแนกได้ครบทุกรูปแบบ

#### **2.1.4.3 Fully-Connected Layer**

เป็นชั้นที่มีค่านำเข้าเป็นข้อมูลจากการกระทำของข้อมูลจากชั้นก่อนหน้าสำหรับนำมาคำนวณเพื่อจำแนกประเภทจากข้อมูลที่ได้มา

### **2.1.3 การเรียนรู้แบบเสริมกำลังเชิงลึก (Deep Reinforcement Learning)**

เป็นวิธีการเรียนรู้แบบเสริมกำลังโดยใช้การเรียนรู้เชิงลึกมาสกัดเพื่อให้ได้ค่าที่เกี่ยวกับการเรียนรู้แบบเสริมกำลัง โดยใช้โครงข่ายประสาทต่อกันจำนวนหลายชั้นมาใช้ในการคำนวนและค่านำเข้าและส่งออกจะเป็นค่าที่อยู่ในการเรียนรู้แบบเสริมกำลัง ซึ่งมักจะถูกใช้กับสภาพแวดล้อมที่เป็นภาพเพื่อเลือกการกระทำไปยังสภาพแวดล้อมที่กำหนด โดยการใช้การเรียนรู้เชิงลึกเป็นโครงสร้างสำหรับการเลือกการกระทำ



รูปที่ 2.4 รูปภาพโครงสร้างของการเรียนรู้แบบเสริมกำลังเชิงลึก

## **2.2 งานวิจัยที่เกี่ยวข้อง**

### **2.2.1 เล่นเกมอาตาริ 2600 โดยใช้การเรียนรู้แบบเสริมกำลัง (Playing Atari with Deep Reinforcement Learning)**

กลุ่ม Deepmind (2556)ได้ทำการสร้างโมเดลสำหรับการเรียนรู้ โดยใช้โครงข่ายคอนโวลูชัน และ ใช้รูปแบบการเรียนรู้ Q-Learning ของการเรียนรู้แบบเสริมกำลัง ซึ่งเรียกว่า Deep Q Networkโดยใช้รูปภาพสำหรับค่านำเข้า และค่าส่งออกเป็นกราฟแสดงผลของรางวัลที่ได้จากการกระทำ และนำมาใช้โดยการนำเกมจากเครื่อง อาตาริ 2600 (Atari 2600) จำนวน 6 เกม และได้นำโมเดลมาใช้กับการเล่นเกม ซึ่งผลที่ได้คือมีทั้งหมด 3 เกมที่มีคะแนนที่มากกว่ามนุษย์ [1]

โครงสร้าง DQN (Deep Q Network) จะรับค่านำเข้าเป็นรูปแต่ละเฟรมเป็นค่านำเข้า ซึ่งประกอบไปด้วย State และ Action ที่เปลี่ยนไปในแต่ละเฟรมภาพ และนำ Q-Learning มาใช้ในการคำนวณเมื่อมีการเปลี่ยนแปลงของสถานะที่มาจากการกระทำที่เกิดขึ้นว่าดีเพียงใด ซึ่งเรียกว่า Q-Value และต่อมาจะทำการเลือกการกระทำที่ดีที่สุดภายให้สถานะที่อยู่เพื่อที่จะไปยังสถานะถัดไปจนกว่ารางวัลที่ได้มากที่สุดหรือใกล้เคียงเป้าหมายที่สุดที่เป็นไปได้ Q-Value จึงเป็นค่าสำคัญมากที่ส่งผลต่อการทำเป้าหมายให้สำเร็จได้และนำโครงข่ายคอนโวลูชันมาใช้รวมกันเพื่อทำการฝึกสอน

### **2.2.2 การเรียนรู้แบบเสริมกำลังโดยใช้วิธีอะซิงโครนัส (Asynchronous Methods for Deep Reinforcement Learning)**

กลุ่ม Deepmind (2559) ได้ทำการสร้างโมเดลแบบใหม่ที่สามารถทำการเรียนรู้แบบคู่ขนานชื่อว่า A3C (Asynchronous Actor-Critic Agents) โดยแบ่งแต่ละกลุ่มโดยใช้อัลกอริทึมของการเรียนรู้แบบเสริมกำลังที่ตางกัน 4 อัลกอริทึมโดยการแบ่งการทำงานแต่ละอัลกอริทึมบนซีพียูของคอมพิวเตอร์แทนที่การ์ดจอ และแสดงผลการทดลองด้วยให้อัลกอริทึมควบคุมการทำงานแขนกล และแก้ปัญหาเกมเขาวงกตแบบ 3 มิติ และได้ทำการเปรียบเทียบกับ DQN โดยการเล่นเกม Atari ซึ่ง A3C มีการเรียนรู้ที่เร็วกว่า DQN และเกมมากกว่าครึ่งหนึ่งที่ทำได้ดี [2]

โครงสร้างของ A3C (Asynchronous Actor-Critic Agents) นั้นมีกระบวนการทำงานคล้ายกับ DQN แต่ต่างกันตรงที่ A3C จะมีการสร้างสภาพแวดล้อมหลักและย่อยโดยแยกกันมากกว่า 1 เอเจนต์ซึ่งจำนวนจะอยู่ที่จำนวนเธรดของซีพียูของคอมพิวเตอร์ เพื่อทำการเรียนรู้ในสถานการณ์ที่ต่างกันเพื่อเก็บประสบการณ์ในการเรียนรู้ และแต่ละส่วนทำการส่งประสบการณ์ส่งกลับไปที่ตัวหลักเพื่อทำการอัพเดทประสบการณ์ไปยังสภาพแวดล้อมหลัก

### **2.2.3 การเข้าถึงค่าประมาณที่ผิดพลาดในกลไกการทำงานแบบ แอคเตอร์-คริติก (Addressing Function Approximation Error in Actor-Critic Methods)**

อัลกอริทึมแบบ Value-Based อย่าง DQN ในบางครั้งมักเกิดอาการที่มีค่า Bias ที่มากกว่าปกติที่ทำให้หาวิธีแก้ปัญหาโดยการแบ่งเป็นสองชุดโดยใช้ Double-Q-Learning เป็นฐานของอัลกอริทึมใหม่ เพื่อลดค่าที่เกินออกมามากผิดปกติและทำการชะลอการอัพเดท Policy เพื่อป้องกันการเกิด Error และพัฒนาประสิทธิภาพของอัลกอริทึม และนำมาประเมินกับสภาพแวดล้อมที่ทาง Openai ได้จัดทำไว้ ซึ่งอัลกอริทึมนี้เรียกว่า TD3 (Twin Delayed Deep Deterministic Policy Gradient) [3]

โครงสร้างของ TD3 (Twin Delayed Deep Deterministic Policy Gradient) เป็นอัลกอริทึมที่พัฒนาต่อมาจาก DDPG (Deep Deterministic Policy Gradient) ซึ่งอัลกอริทึมนี้จะเหมาะกับการควบคุมแบบต่อเนื่อง ตัวอย่าง เช่น การควบคุมการขับรถอัตโนมัติ โดยที่ DDPG เป็นโมเดลที่ดีแต่มีปัญหาอย่างหนึ่งคือ Error จะเพิ่มจนไม่สามารถไปยังจุดที่ดีที่สุดที่เรียกว่า Local Optima TD3 จึงทำมาเพื่อลด Bias ของอัลกอริทึมเดิมโดยการแยก Value Function ที่ต้องการเป็น 2 ส่วน เพื่อมาประเมิน Q-Value แต่ก็มีความเร็วที่น้อยพอสมควร แต่วิธีนี้จะทำให้ Q-Value ไม่มีค่าที่มากเกินไปและทำการอัพเดท Policy ให้น้อยครั้งลงเพื่อไม่ให้เกิด Error กับตัวโมเดลและทำให้เสถียรมากขึ้น

### **2.2.4 มาตรฐานการเรียนรู้แบบเสริมกำลังเพื่อการควบคุมอย่างต่อเนื่อง (Benchmarking Deep Reinforcement Learning for Continuous Control)**

ในงานวิจัยนี้จะพูดถึงเรื่องการนำเอาวิธีพื้นฐานมาใช้กับการเรียนรู้แบบเสริมกำลังได้แก่   
 - REINFORCE(เป็นวิธีพื้นที่ที่ใช้กันทั่วไป)     
 - TNPG(Truncated Natural Policy Gradient)     
 - RWR(Reward-Weighted Regression)    
 - REPS(Relative Entropy Policy Search)   
 - TRPO(Trust Region Policy Optimization)    
 - CEM(Cross Entropy Method )   
 - CMA-ES(Covariance Matrix Adaption Evolution Strategy)    
 - DDPG(Deep Deterministic Policy Gradient)   
 มาเปรียบเทียบกันเพื่อวัดประสิทธิภาพของวิธีที่นำมาใช้ โดยสภาพแวดล้อมที่นำมาทดสอบจะเป็นแบบพื้นฐานที่นำมาใช้กันไปจนถึงเกมในรูปแบบสามมิติ ผลลัพธ์ของการทดลองในงานวิจัยนี้คือ TNPG, TRPO และ DDPG เหมาะสำหรับการฝึกโดยใช้โครงข่ายประสาทเทียมในเชิงลึกเพื่อหาวิธีที่จะไปให้ถึงเป้าหมายได้ดีที่สุด [4]

**2.2.5 การเรียนรู้แบบเสริมกำลังโดยใช้วิธีการ Double Deep Q Network**

เกรก เซอร์มา (Greg Surma) ได้ทำสร้างสร้างการเรียนรู้แบบเสริมกำลังโดยใช้วิธีการ Double Deep Q Network เพื่อพิสูจน์ว่าเอเจนต์ที่ทำงานโดยอัลกอริทึมนั้นสามารถในการแก้ปัญหาในสภาพแวดล้อมแบบต่างๆได้หรือไม่

เขาได้เลือกใช้อัลกอริทึม Double Deep Q Network ซึ่งดีกว่า Deep Q Learning เพราะ DQN มีปัญหาคือเมื่อถึงช่วงค่าแวลู่ฟังก์ชั่นที่ยิ่งมากเกินไป เอเจนต์จะเลือกแต่วิธีที่ดีที่สุดมาเพียงอย่างเดียว และไม่ค้นหาวิธีใหม่เพิ่มเติม

เขาได้ใช้เวลาในการฝึกซ้อมเอเจนต์เป็นเวลา ประมาณ 40 ชั่วโมงบน จีพียู หรือ ประมาณ 90 ชั่วโมงบน ซีพียู Core I7 2.9 Ghz ซึ่งผลที่ได้มีปประสิทธิภาพมากว่าผู้เล่นเกมถึง 1.5 ถึง 2 เท่า

# **บทที่ 3 วิธีการดำเนินงาน**

## **3.1 โปรแกรมหรือซอฟต์แวร์ที่ใช้ในการพัฒนา**

### **3.1.1 ภาษาไพทอน**

สำหรับการเขียนโครงสร้างของงานวิจัย ซึ่งประกอบไปด้วยไลบรารี ดังนี้

#### **3.1.1.1 Gym**

#### เป็นไลบรารีสำหรับการพัฒนาและเปรียบเทียบอัลกอริทึมของการเรียนรู้แบบเสริมกำลัง

#### **3.1.1.2 Gym-retro**

#### เป็นไลบรารีสำหรับการพัฒนาและเปรียบเทียบอัลกอริทึมของการเรียนรู้แบบเสริมกำลัง โดยมีเกมที่อยู่ในช่วงปี 2519 ถึง 2536 ตัวอย่างเช่น Space Invader (2521) จากเครื่อง Atari 2600 และ Sonic The Hedgehog (2534) จากเครื่อง Sega Genesis เป็นต้น

#### **3.1.1.3 Numpy**

เป็นไลบรารีที่ใช้สร้างสูตรการคำนวณที่เกี่ยวข้องกับการเรียนรู้แบบเสริมกำลัง

#### **3.1.1.4 Matplotlib**

เป็นไลบรารีสำหรับการสร้างแผนภูมิสำหรับการวิเคราะห์ข้อมูล

#### **3.1.1.5 Keras**

เป็นไลบรารี Deep learning ที่นำมาใช้ร่วมกับการทำงานของ การเรียนรู้แบบเสริมกำลัง ให้เป็นการทำงานแบบเชิงลึกเพื่อเพิ่มประสิทธิภาพการทำงานให้สูงขึ้น

#### **3.1.1.6 OpenCV**

เป็นไลบรารีที่ใช้ในการประมวลผลด้วยคอมพิวเตอร์แบบเรียลไทม์ ใช้ในการทำงานรูปแบบ Image processing เช่น กล้องจับความเร็วรถ หรือ ระบบสแกนใบหน้า

### **3.1.2** **ไฟล์เกม Kaboom** **ซึ่งเป็นเกมจากเครื่อง Atari 2600**

สำหรับการสร้างสภาพแวดล้อมที่ให้เอเจนต์ได้ทำการฝึกซ้อม ซึ่งมีข้อมูลเกี่ยวกับเกมดังนี้ เป้าหมายของเกมคือรับสิ่งของไม่ให้สิ่งตกลงสู่พื้น ถ้าหากรับไม่ได้จะเสียพลังชีวิต ถ้าหากว่ารับไม่ได้ครบสามครั้งหรือพลังชีวิตของเราหมด หมายความว่าแพ้

## 

## **3.2 ขั้นตอนการดำเนินงาน**

### **3.2.1 สร้างสภาพแวดล้อมสำหรับการนำเอเจนต์อยู่ในพื้นที่ที่กำหนด**

สภาพแวดล้อมที่ผู้จัดทำใช้ไลบรารี Gym-retro ในการนำสภาพแวดล้อมที่เป็นเกมคลาสสิก ซึ่งผู้ใช้ต้องนำไฟล์เกมที่ไลบรารีต้องการ เพื่อสามารถสร้างสภาพแวดล้อมที่ไลบรารีสนับสนุน โดยเกมที่ผู้จัดทำใช้เป็นสภาพแวดล้อมคือเกม Kaboom จากเครื่อง อาตาริ 2600



รูปที่ 3.1 ภาพของเกม Kaboom จากเครื่อง Atari2600

ต้องทำการนำเข้าไฟล์เกมของ Kaboom โดยใช้ชื่อไฟล์ว่า “Kaboom! (Paddle) (CCE).bin” เพื่อสามารถสร้างสภาพแวดล้อมได้ เพราะไลบรารี gym-retro ต้องการไฟล์เกมที่ถูกต้องในการสร้างสภาพแวดล้อม

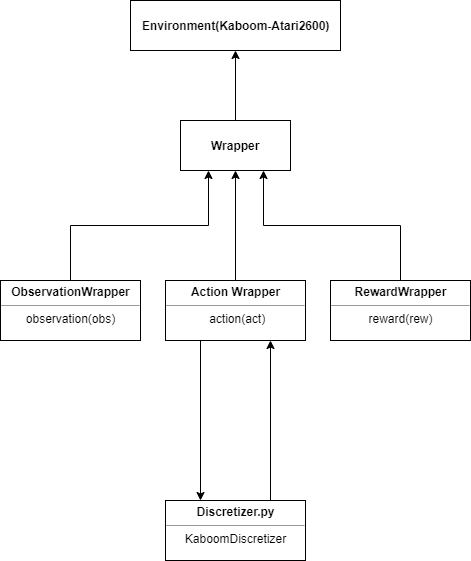
ทำการสร้างสภาพแวดล้อมโดยใช้ฟังก์ชั่น retro.make() จากไลบรารี gym-retro เพื่อสร้างสภาพแวดล้อมและมีลักษณะโครงสร้าง ดังตารางที่ 3.1

|  |  |
| --- | --- |
| ชื่อสภาพแวดล้อม | Kaboom-Atari2600 |
| Observation Space | Box(210, 160, 3) |
| Action Space | MultiBinary(8) |

ตารางที่ 3.1 ลักษณะโครงสร้างของสภาพแวดล้อมของเกม Kaboom

### **3.2.2 คัดกรองการกระทำให้เหลือที่ต้องการตามสภาพแวดล้อม**

เมื่อทำการสร้างสภาพแวดล้อมจะได้จำนวนการกระทำทั้งหมดจำนวน 8 แบบ ได้แก่ ขึ้น, ลง, ซ้าย, ขวา, Button, Select และ null ซึ่งผู้จัดทำต้องการเลือกบางการกระทำที่เหมาะสมกับสภาพแวดล้อมที่กำหนด โดยเกม Kaboom ใช้ปุ่มทั้งหมด 3 ปุ่ม ได้แก่ ซ้าย, ขวา, Button จึงได้สร้างไฟล์ Discretizer.py สำหรับการคัดเลือกการกระทำที่ต้องการ



รูปที่ 3.2 โครงสร้างของ ActionWrapper และ Discretizer.py

ซึ่งได้ทำการเปลี่ยนการกระทำให้เหลือเพียงสามแบบซึ่งเปลี่ยนการกระทำในสภาพแวดล้อมดังตารางที่ 3.2

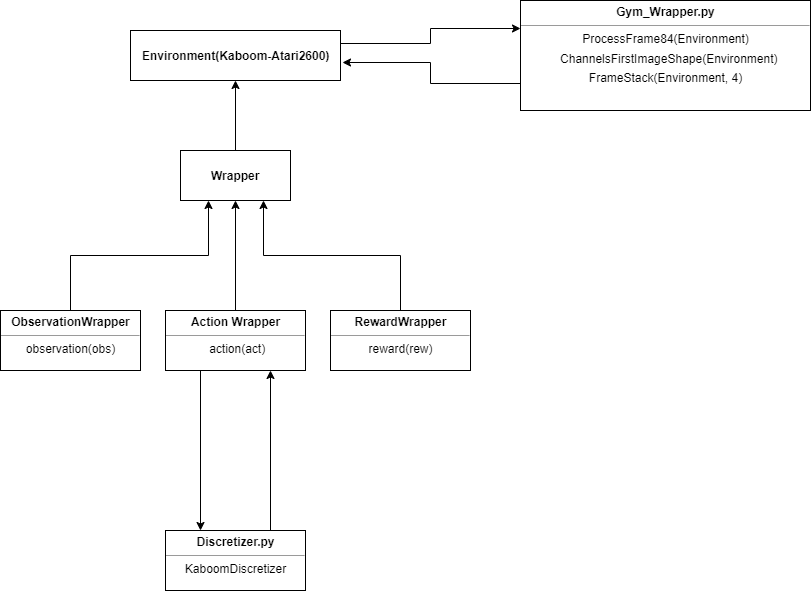
|  |  |
| --- | --- |
| ชื่อสภาพแวดล้อม | Kaboom-Atari2600 |
| Observation Space | Box (210, 160, 3) |
| Action Space | Discrete (8) |

ตารางที่ 3.2 โครงสร้างของสภาพแวดล้อมหลังจากทำการแยกการกระทำที่ต้องการ

### **3.2.3 ปรับขนาดภาพสำหรับการเป็นค่านำเข้าของโครงข่ายคอนโวลูชัน**

หลังจากคัดกรองการกระทำที่ต้องการแล้ว ต้องทำการปรับภาพสำหรับการเป็นข้อมูลสำหรับโครงข่ายแบบคอนโวลูชัน โดยที่ค่านำเข้าคือภาพภายในเกมที่ทำการปรับขนาด และทำให้เป็นสี Grayscale และค่าส่งออกคือ Q-Value ของการกระทำทั้งหมด 3 แบบที่ได้คัดกรองไว้

การแปลงค่าตามทฤษฏีที่เกี่ยวข้องจะลดขนาดของแต่ละภาพขนาด 84x84 พิกเซล จำนวน 4 เฟรมมาต่อกันเพื่อให้ได้ค่ารับเข้าสำหรับโครงข่ายคอนโวลูชั่น โดยที่เฟรมแรกคือ เฟรมภาพปัจจุบันและอีกสามเฟรมคือเฟรมก่อนเฟรมปัจจุบันตามลำดับ



รูปที่ 3.3 โครงสร้างของสภาพแวดล้อมและ Gym\_Wrapper.py

และจะได้โครงสร้างที่พร้อมนำไปใช้กับโครงข่ายคอนโวลูชันดังตารางที่ 3.3

|  |  |
| --- | --- |
| ชื่อสภาพแวดล้อม | Kaboom-Atari2600 |
| Observation Space | Box(4, 84, 84) |
| Action Space | Discrete(8) |

ตารางที่ 3.3 โครงสร้างของสภาพแวดล้อมหลังจากการปรับ Observation สำหรับโครงข่ายคอนโวลูชัน

โดยที่การเตรียมค่าเพื่อที่จะให้กับโครงข่ายแบบคอนโวลูชัน จะมีไฟล์ที่ชื่อว่า Gym\_Wrappers.py ซึ่งมีคลาสที่ประกอบไปด้วย ProcressFrame84, ChannelsFirstImageShape, FrameStack และ ClippedRewardsWrapper

คลาส PreprocessFrame84 เป็นคลาสที่แปลงขนาดของ Observation Space ภาพดังเดิมของสภาพแวดล้อมที่มีขนาด Box(210, 160, 3) กลายเป็น ขนาด Box(84, 84, 1) โดยทำการปรับขนาดและทำภาพให้เป็น Grayscale เพื่อนำไปใช้ในการเก็บเป็นกลุ่มของภาพต่อไป

คลาส ChannelsFirstImageShape เป็นคลาสที่การจัดลำดับภาพที่ได้มาโดยให้ภาพที่ได้มาล่าสุดเป็นภาพแรกของการซ้อนภาพสำหรับการทำเป็นค่านำเข้าของโครงข่ายคอนโวลูชัน

คลาส StackFrame เป็นคลาสที่ทำการซ้อนภาพจำนวน 4 ภาพเพื่อที่ทำการนำมาเป็นค่านำเข้าของโครงข่ายคอนโวลูชัน โดยมีคลาส LazyFrame ที่ช่วยให้การจัดเก็บเกิดขึ้นภายในครั้งเดียวเพื่อประหยัดทรัพยากร

คลาส ClippedRewardsWrapper เป็นการปรับค่ารางวัลที่จะได้จากสภาพแวดล้อมเป็น 1 เมื่อได้รับรางวัลทางบวก (Positive Reward) ซึ่งคือสามารถรับลูกระเบิดที่หล่นลงมา และ -1 เมื่อได้รางวัลทางลบ (Negative Reward) และ 0 เมื่อไม่มีอะไรเกิดขึ้น

เมื่อผ่านคลาสทั้งหมดที่กล่าวมาจะได้สภาพแวดล้อมที่นำไปใช้กับโครงสร้างคอนโวลูชันต่อไป

# **บทที่ 4 ผลการทดลองเบื้องต้น**

จากที่ผู้จัดทำได้ไปศึกษา ค้นคว้า และลองทำการทดลองมา เราได้ทำการทดลองโดยใช้เกม Kaboom ของเครื่องเกง Atari 2600 เป็นเครื่องเกมสมัยก่อน รางวัลที่เราให้กับตัวเอเจนต์มีด้วยกัน 3 รูปแบบ คือ -1 และ 1 โดยที่เมื่อเวลาที่สามารถรับระเบิดได้นั้น ก็จะได้รับรางวัลเป็น 1 คะแนน แต่ถ้าหากว่ารับระเบิดไม่ได้นั้นรางวัลที่ได้ก็จะเป็น -1

ในการทดลองนี้ เราใช้ DDQN อัลกอริทึม ที่มี Batch size ขนาด 32 โดยที่เราจะให้มีการฝึกซ้อม รวมกับการ สุ่มการกระทำ เพื่อที่จะนำข้อมูลจากที่สุ่มการกระทำไปทำการฝึกซ้อม โดยเรากำหนดไว้ทั้งหมดคือ 1,000,000 โดยที่จะแบ่งเป็นเป็นการสุ่มการกระทำ 0.1% หรือก็คือ 10,000 การกระทำ

ในการทดลองนี้ เราได้กำหนดเกม Kaboom จากเครื่อง Atari 2600 มาทำการทดสอบ โดยเราต้องทำการสร้างสภาพแวดล้อมให้กับเอเจนต์เพื่อที่จะเอเจนต์สามารถทำการเล่นเกมได้ หลังจากที่สร้างสภาพแวดล้อมเสร็จ ระบบจะนำสภาพแวดล้อมที่สร้างไปให้เอเจนต์ทำการเรียนรู้ผ่าน โดยในการกระทำชุดแรกจะเป็นการสุ่มการกระทำจำนวนตามที่กำหนด ก็จะนำข้อมูลที่ได้ไปใช้ในการฝึกซ้อม หลังจากทำการเรียนรู้สำเร็จจะแสดงข้อมูลออกมาในรูปแบบกราฟ และ โมเดลสำหรับการนำไปทดสอบต่อไป

## **4.1 การฝึกซ้อม**

ในการเรียนรู้แบบเสริมกำลัง เราจะประเมินความถูกต้องของโมเดลที่นำมาทำการฝึกซ้อม ด้วยวิธีการให้ทดสอบกับตัวเอเจนต์ โดยคะแนนที่เรานำมาทำการสร้างกราฟนั้นเราจะใช้คะแนนเฉลี่ยทุกๆ 10 รอบของการเล่นเกม ถ้าหากว่าคะแนนที่แสดงออกมาดีขึ้น แสดงว่าโมเดลที่เรานำมาทำการฝึกซ้อมให้กับเอเจนต์นั้นมีประสิทธิภาพ โดยปกติแล้วคนจะเล่นคะแนนเฉลี่ยประมาณ 10 คะแนน

กราฟแสดงคะแนนที่เอเจนต์ทำการฝึกซ้อม

## **4.2 ประเมินผลการทดลองที่เกิดขึ้น**

จากผลการทดลองที่เกิดขึ้นข้างต้น แสดงให้เห็นว่าโมเดลที่ถูกนำมาฝึกซ้อมนั้นสามารถเล่นแล้วได้คะแนนที่ไม่เพิ่มขึ้น และไม่มั่นคง แสดงให้เห็นว่าโมเดลที่นำมาใช้ในการทดลองนั้นไม่สามารถนำไปใช้ได้จริง ทั้งนี้สันนิษฐานได้ว่าการปรับตัวแปรยังไม่ดีพอ

# **บทที่ 5 บทสรุป**

# **บรรณาณุกรม**

[1] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).

[2] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International conference on machine learning. 2016.

[3] Fujimoto, Scott, Herke van Hoof, and David Meger. "Addressing function approximation error in actor-critic methods." arXiv preprint arXiv:1802.09477 (2018).

[4] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, “Benchmarking deep reinforcement learning for continuous control,” in International Conference on Machine Learning, 2016, pp. 1329–1338.