การฝึกพฤติกรรมของตัวละครที่ผู้เล่นไม่ได้ควบคุมในเกมโดยวิธีการเรียนแบบเสริมกำลัง

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

สุรเชษฐ์ ใหญ่ธรรมสาร

Surachet Yaitammasan

อัครพล อัครสูริย์

Akarapon Akarasuri

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต
สาขาวิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ภาคเรียนที่ 1 ปีการศึกษา 2562

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

ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2562

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การฝึกพฤติกรรมของตัวละครที่ผู้เล่นไม่ได้ควบคุมในเกมโดยวิธีการเรียนแบบเสริมกำลัง

NON-PLAYER CHARACTER BEHAVIOR TRAINING IN GAME USING REINFORCEMENT LEARNING

ผู้จัดทำ

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

.....อาจารย์ที่ปรึกษา

(ดร. สามารถ หมุดและ)

.....อาจารย์ที่ปรึกษาร่วม

(ดร. สุพัณณดา โชติพันธ์)

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 โดยพื้นฐานความยากง่ายของตัววีดิโอเกมส่วนใหญ่จะมาจากผู้พัฒนาสร้างสคริปต์จะไม่มีความยากง่ายเกินกว่าที่ผู้เล่นทำการเลือก

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 และนำวิธีการเรียนรู้พื้นฐานของการเรียนรู้แบบเสริมกำลังที่ชื่อว่า “Double Deep Q Network ” มาทำการสอนให้คอมพิวเตอร์เรียนรู้และสามารถเล่นได้เอง ผู้จัดต้องการที่จะทราบว่าวิธีการเรียนรู้แบบเสริมกำลังแบบใดเหมาะกับเกมที่นำมาใช้เป็นสภาพแวดล้อมในการเล่น และนำมาเปรียบกับการเล่นกับมนุษย์ว่ามีประสิทธิภาพมากน้อยเพียงใด

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 games is a part of entertainment industries are getting bigger nowadays. Developers are interest to use a Machine Learning into video games. Difficulty in video games are being scripted.

In 2013, Deepmind create a model called “Deep Q Network”. For testing with Atari 2600 games and a result are effectively. Afterward, Deepmind created a “AlphaStar” is an Artificial Intelligence for Starcraft 2. A Real-Time Strategy games. 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 called "Double Deep Q Network". To find which methods are suite an environment and compared with human plays. To show how effective of a model

กิตติกรรมประกาศ

ปริญญานิพธ์นี้สำเร็จลุล่วงได้ด้วยความกรุณาจากดร.สามารถ หมุดและ ดร. สุพัณณดา โชติพันธ์ อาจารย์ที่ปรึกษาโครงงานที่ได้ให้คำแนะนำ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่าง ๆ มาโดยตลอด จนโครงงานเล่มนี้เสร็จสมบูรณ์ ผู้ศึกษาจึงขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณคณาจารย์คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านให้ความรู้กับผู้จัดทำ

ขอขอบคุณเพื่อน ๆ ที่ให้คำปรึกษาการเรื่องการทำงาน และกำลังใจที่ดีเสมอมา

ขอขอบคุณ David Silver ผู้เชี่ยวชาญทางด้านการเรียนรู้แบบเสริมกำลัง ที่มอบแนวคิด ทฤษฎี และให้ความรู้เกี่ยวกับ การเรียนรู้แบบเสริมกำลัง

และความดีอันเกิดจากการศึกษาค้นคว้าครั้งนี้ ผู้เขียนขอมอบแด่ผู้ที่มีความประสงค์ที่จะเรียนรู้เกี่ยวกับการเรียนรูแบบเสริมกำลัง ผู้เขียนมีความซาบซึ้งในความกรุณาอันยิ่งใหญ่จากทุกท่านที่ได้กล่าวนามมาและขอกราบขอบพระคุณมา ณ โอกาสนี้

นาย สุรเชษฐ์ ใหญ่ธรรมสาร
นาย อัครพล อัครสูริย์

สารบัญ

สารบัญรูป

หน้า

รูปที่

สารบัญตาราง

หน้า

ตารางที่

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 ทำลายสิ่งปลูกสร้างที่อยู่ในฐานของทีมอีกฝ่าย แต่จนกว่าจะไปถึงเป้าหมายนั้นจะมีรายละเอียดเล็กน้อยหรือเป้าหมายย่อยที่ทำให้สามารถสำเร็จเป้าหมายของเกมได้

ทางผู้จัดทำสนใจที่ทำการเรียนรู้การเรียนรู้แบบเสริมกำลังด้วยมีสภาพแวดล้อมให้กับปัญญาประดิษฐ์ด้วยวิดีโอเกม แต่ด้วยข้อจำกัดของอุปกรณ์ ผู้จัดทำนำเกมที่มีความซับซ้อนน้อยลงโดยเป็นเกมที่อยู่ในยุคเริ่มต้นของอุตสากรรมวิดีโอเกมที่มีความละเอียดและความซับซ้อนที่น้อยลงเพื่อนำมาศึกษาหลักการและอัลกอริทึมที่เกี่ยวกับการเรียนรู้แบบเสริมกำลัง

1.2 วัตถุประสงค์

1. เพื่อพัฒนาระบบปัญญาประดิษฐ์กับวีดิโอเกมด้วยวิธีการ Reinforcement Learning
2. เพื่อพัฒนาองค์ความรู้ด้านการ Reinforcement Learning
3. ศึกษาหลักการและโครงสร้างของการเรียนรู้แบบเสริมกำลังผ่านวิดีโอเกม

1.3 ขอบเขตของโครงงาน

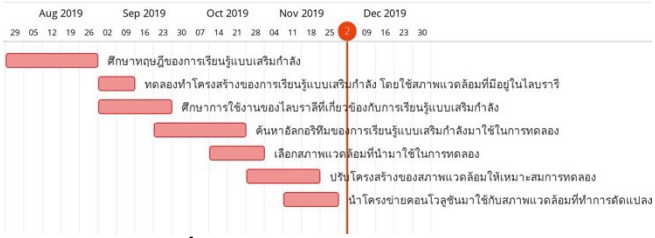
1. ออกแบบวิธีการด้วยวิธีการเรียนรู้แบบเสริมกำลังสำหรับพัฒนาพฤติกรรมของ NPC
2. เปรียบเทียบผลของ NPC ที่ใช้อัลกอริทึมที่ต่างกันของวิธีการเรียนรู้แบบเสริมกำลัง
3. วิเคราะห์ผลของเกมที่นำ NPC ที่ผ่านการพัฒนาโดยอัลกอริทึมต่าง ๆ ของการเรียนรู้แบบเสริมกำลัง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้แบบจำลองจากการฝึกสอนให้ปัญญาประดิษฐ์ที่สามารถทำคะแนนได้ดีในเกม Kaboom
2. ได้ทราบวิธีในการฝึกสอนให้กับปัญญาประดิษฐ์

1.5 แผนการดำเนินงาน

ในการดำเนินโครงงานในช่วงเวลา 1 เดือนแรกได้ใช้เวลาในการค้นหาข้อมูลที่เกี่ยวข้องกับการเรียนรู้แบบเสริมกำลังต่อมาได้ทำการศึกษาไลบรารีที่เกี่ยวข้องกับการเรียนรู้แบบเสริมกำลังละได้ทดลองใช้และทำการเลือกสภพแวดล้อมที่นำมาใช้ในการทดลอง และนำอัลกอลิทึมหนึ่งอัลกอริทึมนำมาทดลองที่จะนำมาเป็นโครงสร้างหลักในการนำอัลกอริทึมอื่น ๆที่สนใจมาทำการทดลองต่อไป



รูปที่ 1.1 แผนภาพการดำเนินงาน

บทที่ 2
ทฤษฎี งานวิจัย และเครื่องมือที่เกี่ยวข้อง

2.1 ทฤษฎี

2.1.1 การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning)

การเรียนรู้แบบเสริมกำลังเป็นส่วนหนึ่งของการเรียนรู้ด้วยเครื่อง (Machine Learning) ที่ทำการโดยนำปัญญาประดิษฐ์นำมาอยู่ภายใต้สภาพแวดล้อมหนึ่งที่ปัญญาประดิษฐ์ทำการตัดสินใจในการกระทำหนึ่ง ที่มาจากการสุ่มหรือ เลือกการกระทำจากข้อมูลที่อยู่ภายใต้สิ่งแวดล้อมที่กำหนด ซึ่งปัญญาประดิษฐ์ของการเรียนรู้แบบเสริมกำลังมีเป้าหมายคือ เลือกการกระทำที่ทำให้รับรางวัลที่ดีที่สุดในการแก้ปัญหาหนึ่ง ผ่านการลองผิดลองถูกของตัวปัญญาประดิษฐ์

การเรียนรู้แบบเสริมกำลังส่วนใหญ่จะถูกใช้ในอุตสาหกรรมหุ่นยนต์ และอุตสาหกรรมวิดีโอเกม เช่น ใช้การเรียนรู้แบบเสริมกำลังสร้างปัญญาประดิษฐ์ในการเล่นเกมคอมพิวเตอร์ Starcraft 2[1] หรือ สร้างปัญญาประดิษฐ์ควบคุมการทำงานของแขนกล ซึ่งองค์ประกอบของการเรียนรู้แบบเสริมกำลังมีทั้งหมด 5 ส่วน [2]

2.1.1.1 เอเจนต์ (Agent)

ปัญญาประดิษฐ์ที่อยู่ภายใต้สภาพแวดล้อมและการทำงานของการเรียนรู้แบบเสริมกำลัง ซึ่งภายในเอเจนต์หนึ่งตัวจะมีส่วนประกอบภายใน 1 ประเภทหรือมากกว่า ซึ่งมีทั้งหมด 3 องค์ประกอบดังนี้

- **Policy**

เป็นกฎเกณฑ์ของวิธีการที่จะให้ไปถึงเป้าหมายที่ต้องการเมื่ออยู่ในสถานะที่ต่างกันออกไปเพื่อให้วิธีที่ดีที่สุดในการทำเป้าหมาย

- **Value Function**

เป็นค่าที่ใช้วัดผลจากการกระทำในสถานะต่าง ๆ เพื่อวัดผลว่าถ้าทำการกระทำหนึ่ง ณ สถานะปัจจุบันให้ผลดีต่อรางวัลในอนาคตอย่างไร

- **Model**

โมเดลเป็นการทำนายว่าในสภาพแวดล้อมจะเกิดอะไรขึ้นต่อไปทั้งสถานะและรางวัลที่จะได้จากการกระทำ

2.1.1.2 สภาพแวดล้อม (Environment)

เป็นพื้นที่ทำการนำปัญญาประดิษฐ์ทำภารกิจในสภาพแวดล้อมที่กำหนด

2.1.1.3 สถานะ (State)

เป็นสถานะของสภาพแวดล้อมในช่วงเวลาต่าง ๆที่ปัญญาประดิษฐ์สามารถรับรู้เพื่อตัดสินใจเลือกการกระทำในแต่ละช่วงเวลา

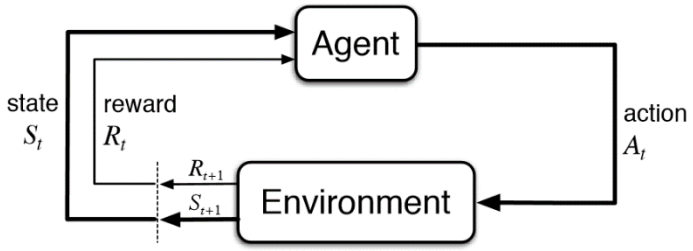
2.1.1.4 การกระทำ (Action)

เอเจนต์จะเลือกการกระทำที่ทำการตัดสินใจจากสถานะก่อนหน้าเข้าไปยังสภาพแวดล้อมเพื่อแสดงสถานะถัดไปและรางวัลที่ได้จากการเลือกการกระทำ

2.1.1.5 รางวัล (Reward)

เป็นรางวัลที่ได้จากการกระทำเพื่อแสดงว่าการกระทำที่เลือกไปดีต่อกับสภาพแวดล้อมและเป้าหมายของสภาพแวดล้อมได้ดีเพียงใด

โดยโครงสร้างของการเรียนรู้แบบเสริมกำลังจะมีวงจรเป็นการวนซ้ำของการกระทำของเอเจนต์ไปยังสภาพแวดล้อมและผลที่เกิดขึ้นและรางวัลที่ได้จากการกระทำไปยังเอเจนต์ ดังรูปที่ 2.1



รูปที่ 2.1 โครงสร้างการทำงานของการเรียนรู้แบบเสริมกำลัง

ซึ่งการแก้ปัญหาที่ให้การเรียนรู้แบบเสริมกำลังส่วนใหญ่มักจะแก้ไขด้วยหลักการที่ชื่อว่ากระบวนการตัดสินใจของมาร์คอฟ [3] (Markov Decision Process) ซึ่งมีคุณสมบัติดังนี้

สถานะใด ๆ สถานะหนึ่งจะเป็นสถานะมาร์คอฟได้ก็ต่อเมื่อ สถานะปัจจุบันนั้นส่งผลถึงสถานะในอนาคต เท่ากับสถานะปัจจุบันและสถานะก่อนหน้าส่งผลกับสถานะในอนาคต ซึ่งทำให้ไม่ต้องสนใจสถานะในอดีตอีกต่อไป

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, \ldots, S_t] \text{ ; เมื่อ } S_t \text{ เป็นสถานะมาร์คอฟ} \quad (1)$$

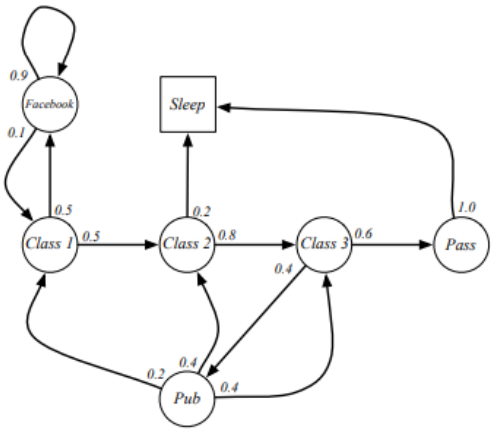
State Transition Matrix เป็นเมทริกซ์ของความน่าจะเป็นของสถานะที่อยู่ไปยังสถานะต่อไปซึ่งแต่ละจุดจะมีความน่าจะเป็นดังสมการ

$$P_{ss'} = P[S_{t+1} = s' | S_t = s] \quad (2)$$

ซึ่งแต่ละจุดสามารถรวมเป็นเมทริกซ์ได้ดังนี้

$$P = \begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix} \quad (3)$$

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



$$\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\ \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

รูปที่ 2.2 โครงสร้างของห่วงโซ่มาร์คอฟ

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

$$S_1, S_2, S_3, \ldots, S_T \quad (4)$$

โดยการวัดจากโมเดลที่สร้างมาจากการเรียนรู้แบบเสริมกำลังจะมีผลทั้งหมด 2 แบบคือ รางวัลโดยรวมในแต่ละเอพิโซด (Episodic Return) และ ค่าแวลูฟังก์ชัน (Value Function) ซึ่งรางวัล
รางวัลโดยรวมในแต่ละ Episode จะเป็นรางวัลที่คาดหวังปัจจุบันที่อยู่ในภายใต้สภาพแวดล้อมที่เอเจนต์ทำงานอยู่ภายในที่อยู่ในช่วงเวลานั้น ตัวอย่าง เช่น ถ้าสภาพแวดล้อมเป็นวิดีโอเกมรางวัลของการเล่นเกมคือได้คะแนนเพิ่มขึ้น หรือถ้าทำการฝึกแขนกล รางวัลคือการทำงานสำเร็จในแต่ละครั้ง เป็นต้น

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (5)$$

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

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

State-Value Function จะเป็นผลรวมของรางวัลที่สถานะไปยังสถานะใหม่ตาม Policy π เพื่อดูว่าการเคลื่อนไปยังสถานะใหม่นั้นมีผลดีเพียงใด

$$V_\pi(S) = E_\pi[G_t | S_t = s] \quad (6)$$

Action-Value Function จะเป็นผลรวมของรางวัลที่สถานะที่เลือกการกระทำที่นำพาไปยังสถานะใหม่ตาม Policy π เพื่อดูว่าการเคลื่อนไปยังสถานะใหม่นั้นมีผลดีเพียงใด

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] \quad (7)$$

เมื่อเอเจนต์ได้ไปอยู่ในสภาพแวดล้อมหนึ่งเป็นครั้งแรก แล้วจะรู้ได้อย่างไรว่าเลือกการกระทำนี้แล้วจะส่งผลดีต่อเอเจนต์หรือไม่ หรือเมื่อทำการหาการกระทำแล้วได้ผลลัพธ์มาจำนวนหนึ่ง แล้วจะรู้ได้อย่างไรว่าการกระทำนี้เหมาะกับสถานะนี้แล้วหรือไม่ โดยมีสิ่งที่เรียกว่า ε-Greedy (Epsilon Greedy)[4] เพื่อเลือกว่าจะทำการค้นหาวิธีใหม่ หรือเลือกใช้วิธีที่ดีที่สุดในสถานะนั้น

โดยค่า Epsilon Greedy มีค่าอยู่ระหว่าง 0-1 ซึ่งยิ่งมีค่าเข้าใกล้ 1 จะทำให้ทำการค้นหาวิธีการใหม่ ๆ แต่ถ้ายิ่งน้อย เอเจนต์จะเลือกวิธีการที่ดีที่สุดมาใช้ในสถานะนั้น

รูปที่ 2.3 กราฟแสดงหลักการ Exploration และ Exploitaion

2.1.2 Gym และ Gym-retro

Gym [4] เป็นไลบรารีโอเพนซอร์ซที่ทาง OpenAI สร้างขึ้นเพื่อให้ผู้ที่มีความสนใจในการพัฒนาในการเรียนรู้แบบเสริมกำลัง เพื่อพัฒนาและเปรียบเทียบอัลกอริทึมของการเรียนรู้แบบเสริมกำลังภายใต้สภาพแวดล้อมต่าง ๆ สภาพแวดล้อมที่ทาง Gym มีให้ก็มีด้วยกันหลายรูปแบบด้วยกัน ไม่ว่าจะเป็นเกมที่เป็นข้อความ หรือวีดิโอเกม หรือไปจนถึงการฝึกสอนหุ่นยนต์ โดยเป้าหมายเพื่อให้เอเจนต์(ปัญญาประดิษฐ์)สามารภทำภารกิจได้ลุล่วงตามสภาพแวดล้อมที่กำหนด

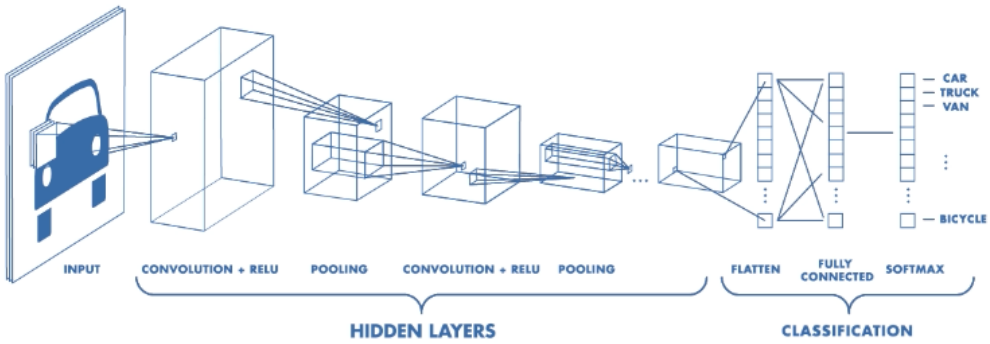
Gym-retro [5] คือไลบรารีโอเพนซอร์ซที่ทาง OpenAI สร้างขึ้นโดยมีรากฐานเดียวกับ Gym คือ ให้ผู้ที่สนใจพัฒนาในการเรียนรู้แบบเสริมกำลัง แต่ว่ามีความแตกต่างที่สภาพแวดล้อมของ Gym-retro จะเป็นเกมคลาสสิคที่เกิดในยุคแรกของอุตสาหกรรมวีดิโอเกม(ในช่วงประมาณ ค.ศ.1970 - ค.ศ.1995) เช่น Space Invader, Super Mario Bros, Sonic The Hedgehog เป็นต้น และเกมที่อยู่ในช่วงเวลาเดียวกัน

2.1.3 เกม Kaboom

เกม Kaboom เป็นเกมจากเครื่อง Atari 2600 เป็นเกมในยุคเริ่มต้นของอุตสาหกรรมเกมซึ่งมีความละเอียดและความซับซ้อนของเกมที่น้อย ภายในเกมจะมีโจรปล่อยระเบิดลงมาเพื่อให้ผู้เล่นรับระเบิด เมื่อรับสำเร็จจะได้รับคะแนน 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 องค์ประกอบของโครงข่ายคอนโวลูชันมีดังนี้



รูป 2.4 โครงสร้างของโครงข่ายประสาทแบบคอนโวลูชัน

2.1.4.1 Convolutional Layer

เป็นชั้นที่ทำการสแกนค่ารับเข้าซึ่งเป็นรูปภาพ เพื่อแยกองค์ประกอบของรูป เช่น สี รูปทรง ขอบของภาพ

2.1.4.2 Pooling Layer

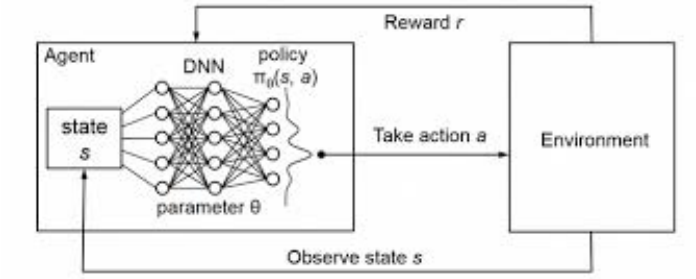
เป็นการลดขนาดของข้อมูลมีขนาดเล็กลงโดยที่รายละเอียดยังคงเดิม ซึ่งทั้ง Convolutional Layer และ Pooling Layer จะทำงานคู่กันซึ่งจะทำงานหลายครั้งเพื่อจำแนกได้ครบทุกรูปแบบ

2.1.4.3 Fully-Connected Layer

เป็นชั้นที่มีค่านำเข้าเป็นข้อมูลจากการกระทำของข้อมูลจากชั้นก่อนหน้าสำหรับนำมาคำนวณเพื่อจำแนกประเภทจากข้อมูลที่ได้มา

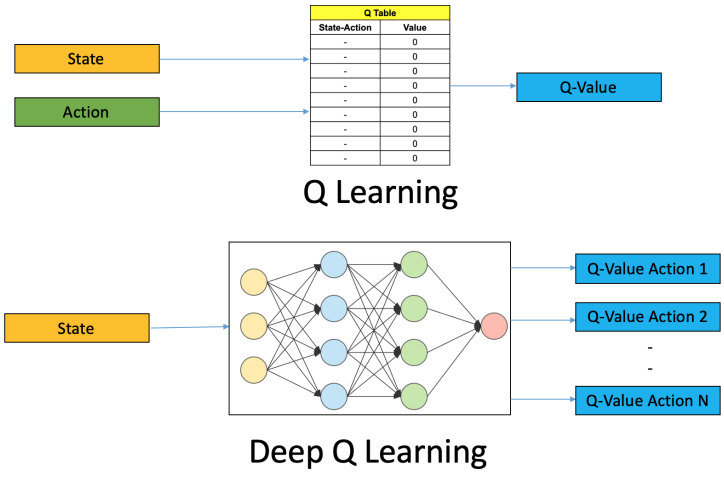
2.1.5 การเรียนรู้แบบเสริมกำลังเชิงลึก (Deep Reinforcement Learning)

เป็นวิธีการเรียนรู้แบบเสริมกำลังโดยใช้การเรียนรู้เชิงลึกมาสกัดเพื่อให้ได้ค่าที่เกี่ยวกับการเรียนรู้แบบเสริมกำลัง โดยใช้โครงข่ายประสาทต่อกันจำนวนหลายชั้นมาใช้ในการคำนวนและค่านำเข้าและส่งออกจะเป็นค่าที่อยู่ในการเรียนรู้แบบเสริมกำลัง ซึ่งมักจะถูกใช้กับสภาพแวดล้อมที่เป็นภาพเพื่อเลือกการกระทำไปยังสภาพแวดล้อมที่กำหนด โดยการใช้การเรียนรู้เชิงลึกเป็นโครงสร้างสำหรับการเลือกการกระทำ



รูปที่ 2.5 รูปภาพโครงสร้างของการเรียนรู้แบบเสริมกำลังเชิงลึก

โดยอัลกอริทึมของการเรียนรู้แบบเสริมกำลังที่ใช้รวมกับการเรียนรู้เชิงลึก คือ Deep Q Network [6] ซึ่งเป็นอัลกอริทึมที่นำโครงข่ายคอนโวลูชันมาประยุกต์ใช้กับ ค่า Q-Value ที่อยู่ในรูปแบบของตารางเพื่อใช้ในการตัดสินใจ เปลี่ยนมาใช้โครงข่ายประสาทเป็นตัวคำนวณการตัดสินใจเพื่อเลือกการกระทำที่ดีมากระทำเพื่อให้ได้เป้าหมายที่ต้องการ ดังรูปที่ 2.5



รูปที่ 2.6 ตาราง Q-Value (บน) Deep Q Network (ล่าง)

2.2 งานวิจัยที่เกี่ยวข้อง

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

กลุ่ม Deepmind ได้ทำการสร้างโมเดลสำหรับการเรียนรู้ โดยใช้โครงข่ายคอนโวลูชัน และใช้รูปแบบการเรียนรู้ Q-Learning ของการเรียนรู้แบบเสริมกำลัง ซึ่งเรียกว่า Deep Q Network โดยใช้รูปภาพสำหรับค่านำเข้า และค่าส่งออกเป็นกราฟแสดงผลของรางวัลที่ได้จากการกระทำ และนำมาใช้โดยการนำเกมจากเครื่อง อาตาริ 2600 (Atari 2600) จำนวน 6 เกม และได้นำโมเดลมาใช้กับการเล่นเกม ซึ่งผลที่ได้คือมีทั้งหมด 3 เกมที่มีคะแนนที่มากกว่ามนุษย์ [7]

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

2.2.2 การเรียนรู้แบบเสริมกำลังโดยใช้วิธีอะซิงโครนัส (Asynchronous Methods for Deep Reinforcement Learning)

กลุ่ม Deepmind ได้ทำการสร้างโมเดลแบบใหม่ที่สามารถทำการเรียนรู้แบบคู่ขนานชื่อว่า A3C (Asynchronous Actor-Critic Agents) โดยแบ่งแต่ละกลุ่มโดยใช้อัลกอริทึมของการเรียนรู้แบบเสริมกำลังที่ตางกัน 4 อัลกอริทึมโดยการแบ่งการทำงานแต่ละอัลกอริทึมบนซีพียูของคอมพิวเตอร์แทนที่การ์ดจอ และแสดงผลการทดลองด้วยให้อัลกอริทึมควบคุมการทำงานแขนกล และแก้ปัญหาเกมเขาวงกตแบบ 3 มิติ และได้ทำการเปรียบเทียบกับ DQN โดยการเล่นเกม Atari ซึ่ง A3C มีการเรียนรู้ที่เร็วกว่า DQN และเกมมากกว่าครึ่งหนึ่งที่ทำได้ดี [8]

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 เป็นฐานของอัลกอริทึมใหม่เพื่อลดค่าที่เกินออกมามากผิดปกติและทำการชะลอการอัพเดท Policy เพื่อป้องกันการเกิด Error และพัฒนาประสิทธิภาพของอัลกอริทึม และนำมาประเมินกับสภาพแวดล้อมที่ทาง Openai ได้จัดทำไว้ ซึ่งอัลกอริทึมนี้เรียกว่า TD3 (Twin Delayed Deep Deterministic Policy Gradient) [9]

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 การเรียนรู้แบบเสริมกำลังโดยใช้วิธีการ Double Deep Q Network

เกรก เซอร์มา (Greg Surma) ได้ทำสร้างสร้างการเรียนรู้แบบเสริมกำลังโดยใช้วิธีการ Double Deep Q Network เพื่อพิสูจน์ว่าเอเจนต์ที่ทำงานโดยอัลกอริทึมนั้นสามารถในการแก้ปัญหาในสภาพแวดล้อมแบบต่าง ๆ ได้หรือไม่ [10]

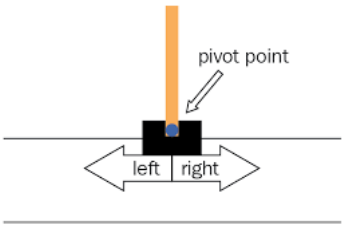
จึงได้เลือกใช้อัลกอริทึม Double Deep Q Network ซึ่งดีกว่า Deep Q Learning เพราะ DQN มีปัญหาคือเมื่อถึงช่วงค่าแวลูฟังก์ชั่นที่ยิ่งมากเกินไป เอเจนต์จะเลือกแต่วิธีที่ดีที่สุดมาเพียงอย่างเดียว และไม่ค้นหาวิธีใหม่เพิ่มเติม

เขาได้ใช้เวลาในการฝึกสอนเอเจนต์เป็นเวลา ประมาณ 40 ชั่วโมงบน จีพียู หรือ ประมาณ 90 ชั่วโมงบน ซีพียู Core I7 2.9 กิกะเฮิร์ทซ์ ซึ่งผลที่ได้มีปประสิทธิภาพมากว่าผู้เล่นเกมถึง 1.5 ถึง 2 เท่า

2.3 โปรแกรมหรือซอฟต์แวร์ที่ใช้ในการพัฒนา

2.3.1 ภาษาไพทอน สำหรับการเขียนโครงสร้างของโครงงาน ซึ่งประกอบไปด้วยไลบรารี ดังนี้

1) **Gym** เป็นไลบรารีสำหรับการพัฒนาและเปรียบเทียบอัลกอริทึมของการเรียนรู้แบบเสริมกำลัง โดยเป็นส่วนหลักที่นำมาเป็นโครงสร้างของการพัฒนาการเรียนรู้แบบเสริมกำลัง ใน gym จะมีสภาพแวดล้อมพื้นฐานสำหรับการพัฒนาการเรียนรู้แบบเสริมกำลังมาให้ใช้งานเบื้องต้น เช่น CartPole ที่เป็นที่นิยมในการศึกษาการเรียนรู้แบบเสริมกำลังในเริ่มแรกไปจนถึง Robotics สำหรับการจำลองร่างกายให้กับหุ่นยนต์



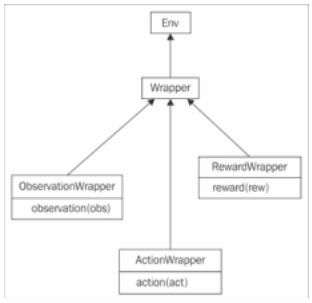
รูปที่ 2.7 ตัวอย่างสภาพแวดล้อมใน Gym

โครงสร้างของ Gym ประกอบไปด้วยส่วนประกอบหลักๆ 3 ส่วน

1.Observation Wrapper คือ สภาพแวดล้อม ณ จุดๆนั้นที่ปัญญาประดิษฐ์สามารถที่จะรับรู้และกระทำสิ่งต่างๆได้ เช่น เรายืนอยู่หน้าทางแยกที่มีอยู่สองทาง Observation จะเป็นภาพ ณ เวลานั้นที่เรารับรู้ว่ามีทางแยกที่สามารถเดินทางไปต่อได้

2. Action Wrapper คือ การกระทำใน Observation ที่เราสามารถทำได้ เช่น เราต้องการที่จะให้อาหารกับสุนัข Action จะเป็นการกระทำที่เป็นไปได้ทั้งหมด ไม่ว่าเราจะให้อาหารกับสุนัขด้วยการป้อนหรือจะใส่ในชามข้าว หรืออื่นๆที่จะเกิดขึ้นได้

3. Reward Wrapper คือ รางวัลที่ได้จากการกระทำเมื่อเรากระทำสำเร็จ เช่น เราฝึกสุนัขและเราขอมือ ถ้าสุนัขกระทำโดยการยื่นมือมาให้กับเราตอบ รางวัลที่ได้ก็จะเป็นผลดีเช่น ได้รับขนม แต่ถ้าเกิดเราขอมือสุนัขแต่สุนัขกัดมือของเรา สุนัขก็จะได้รางวัลที่เป็นผลเสียเช่น การดุ หรือไม่ให้ขนม



รูปที่ 2.8 โครงสร้างในการสร้างสภาพแวดล้อมของ Gym

2) **Gym-retro** เป็นไลบรารีสำหรับการพัฒนาและเปรียบเทียบอัลกอริทึมของการเรียนรู้แบบเสริมกำลัง โดยมีเกมที่อยู่ในช่วงปี 2519 ถึง 2536 ตัวอย่างเช่น Space Invader (2521) จากเครื่อง Atari 2600 และ Sonic The Hedgehog (2534) จากเครื่อง Sega Genesis เป็นต้น ในปี 2561 ได้มีการการแข่งขันเกี่ยวกับการเรียนรู้แบบเสริมกำลังโดยใช้ Gym-retro มาใช้ในการสร้างสภาพแวดล้อมเพื่อฝึกสอนให้กับปัญญาประดิษฐ์ เป็นการแข่งขันฝึกสอนปัญญาประดิษฐ์วิธีใดมีประสิทธิภาพมากที่สุด โดยในโครงงานจะใช้ไลบรารีในการสร้างสภาพแวดล้อม และการเรียนรู้แบบเสริมกำลังจะใช้จากไลบรารี Gym เป็นหลัก

รูปที่ 2.9 ตัวอย่างสภาพแวดล้อมใน Gym-Retro

3) **Numpy** เป็นไลบรารีที่ใช้สร้างสูตรการคำนวณที่เกี่ยวข้องกับคณิตศาสตร์ ภายในโครงงานนี้จะทำการเก็บค่าของการเรียนรู้แบบเสริมกำลัง และการจัดเก็บและดัดแปลงข้อมูลให้อยู่ในรูปแบบของเมทริกซ์ และแปลงรูปภาพของสภาพแวดล้อมเป็นอาเรย์ขนาดใหม่ที่ใช้สำหรับการนำไปประมวลผล

4) **Matplotlib** เป็นไลบรารีสำหรับการสร้างแผนภูมิสำหรับการวิเคราะห์ข้อมูล นำมาใช้สำหรับการแสดงผลการทดลองออกมาทางรูปแบบของแผนภูมิของโครงงานในหัวข้อต่าง ๆ เพื่อนำมาสรุปผลการทดลอง

5) **Keras** เป็นไลบรารี Deep learning ที่นำมาใช้ร่วมกับการทำงานของ การเรียนรู้แบบเสริมกำลัง ให้เป็นการทำงานแบบเชิงลึกเพื่อเพิ่มประสิทธิภาพการทำงานให้สูงขึ้นภายในโครงงานได้นำมาใช้สร้างโครงข่ายประสาทแบบคอนโวลูชันเพื่อนำมาประมวลผลของเอเจนต์

6) **OpenCV** เป็นไลบรารีที่ใช้ในการประมวลผลด้วยคอมพิวเตอร์แบบเรียลไทม์ ใช้ในการทำงานรูปแบบ Image processing เช่น กล้องจับความเร็วรถ หรือ ระบบสแกนใบหน้า โครงงานนี้ได้นำไลบรารีนี้สำหรับการแปลงภาพของสภาพแวดล้อมที่เป็นสี แปลงให้เป็นภาพขาวดำเพื่อลดขนาดของข้อมูลที่ใช้ประมวลผล

3.1.2 ไฟล์เกม Kaboom ซึ่งเป็นเกมจากเครื่อง Atari 2600

สำหรับการสร้างสภาพแวดล้อมที่ให้เอเจนต์ได้ทำการฝึกสอน ซึ่งมีข้อมูลเกี่ยวกับเกมดังนี้ เป้าหมายของเกมคือรับสิ่งของไม่ให้สิ่งตกลงสู่พื้น ถ้าหากรับไม่ได้จะเสียพลังชีวิต ถ้าหากว่ารับไม่ได้ครบสามครั้งหรือพลังชีวิตของเราหมด หมายความว่าแพ้ โดยที่ไฟล์ที่นำมาใช้ที่มีชื่อว่า “Kaboom! (Paddle) (CCE).bin” ซึ่งเป็นไฟล์ที่ใช้ตัวแทนตลับเกมของจริง ซึ่งนำมาใช้กับโปรแกรมจำลองการเล่นเกม และเป็นไฟล์ที่ไลบรารี Gym-Retro รองรับในการสร้างสภาพแวดล้อมของเกม

บทที่ 3

วิธีการพัฒนาโปรแกรม

3.1 เลือกสภาพแวดล้อมที่นำมาใช้

รูปที่ 3.1 ภาพของเกม Kaboom จากเครื่อง Atari2600

เกม kaboom เป็นเกมที่เผยแพร่จากบริษัท Activision ในปี 1981 สำหรับเครื่อง Atari2600 ที่ออกแบบโดย Larry Kaplan ที่ได้รับการตอบรับเป็นอย่างดี และมียอดขายได้มากกว่าหนึ่งล้านตลับในปี 1983 ซึ่งมีรายละเอียดของเกมเป็นดังนี้

ชื่อเกม : Kaboom

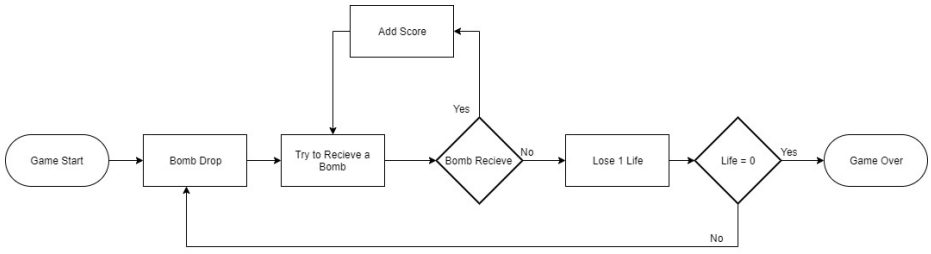
ประเภท : แอ็คชัน

จำนวนผู้เล่น : 1 ผู้เล่น

รูปแบบการเล่น : ในการเล่นหนึ่งเกม เมื่อเริ่มต้นจะมีโจรที่อยู่ด้านบนของเกมปล่อยสิ่งของมาให้ผู้เล่นได้ทำการรับ เมื่อทำการรับเสร็จจะได้คะแนน และถ้าไม่สามารถรับสิ่งที่ตกลงมาได้จะเสียพลังหนึ่งชีวิต

เป้าหมาย : ผู้เล่นไม่สามารถรับสิ่งของได้ทั้งหมดสามครั้งเป็นอันจบเกม

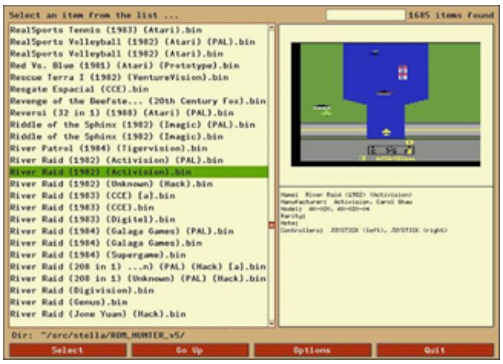
การจบเกม : รับสิ่งของเพื่อให้ได้คะแนนมากที่สุดเท่าที่เป็นไปได้



รูปที่ 3.2 ผังการดำเนินของเกม Kaboom

3.2 การสร้างสภาพแวดล้อม

การสร้างสภาพแวดล้อมของเกมคลาสสิคต้องใช้ไลบรารี Gym-Retro สำหรับการสร้าง ซึ่งภายใน Gym-Retro จะทำการสร้างสภาพแวดล้อมที่เป็นเกมประเภทคลาสสิคโดยที่เกมคลาสสิคต่าง ๆ จะอยู่ภายใต้การทำงานของโปรแกรมจำลองการเล่นของเกมในแต่ละแพลตฟอร์ม ซึ่งโดยเกมที่มาจากเครื่องของ Atari 2600 ตัวเกมที่สามารถสร้างเป็นสภาพแวดล้อม นั้นอยู่ภายใต้ของโปรแกรมจำลองการเล่นเกมที่ชื่อว่า Stella โดยที่เกมที่อยู่ในเครื่อง Atari 2600 มีเกมที่มีความนิยมหลาย ๆ เกมเช่น Space Invader, Q*Bert, Seaquest และ Kaboom เป็นต้น



รูปที่ 3.3 Stellaโปรแกรมจำลองการเล่นเกมเครื่อง Atari2600

โดยการสร้างสภาพแวดล้อมต้องใช้ไฟล์เกมที่ถูกต้องตามที่ Gym-Retro ตามที่กำหนดซึ่งซึ่งมีค่า Hash สำหรับการกำหนดว่าไฟล์เกมที่นำมาใช้ตรงกับไฟล์ที่จะสร้างสภาพแวดล้อมของ Gym-Retro โดยวิธีการนำเข้าไฟล์เกมต้องใช้คำสั่ง “python3 -m retro.import” ณ ตำแหน่งของไฟล์เกมที่อยู่ ไฟล์ที่ถูกต้องชื่อของสภาพแวดล้อมจะอยู่ในรูปแบบของ “ชื่อเกม-ชื่อแพลตฟอร์ม” ดังรูป

```
D:\reinforcement-learning\Game rom>python -m retro.import
Importing 2 potential games...
Importing Kaboom-Atari2600
Imported 1 games
```

รูปที่ 3.4 วิธีการนำเข้าไฟล์เกม สำหรับการสร้างสภาพแวดล้อม

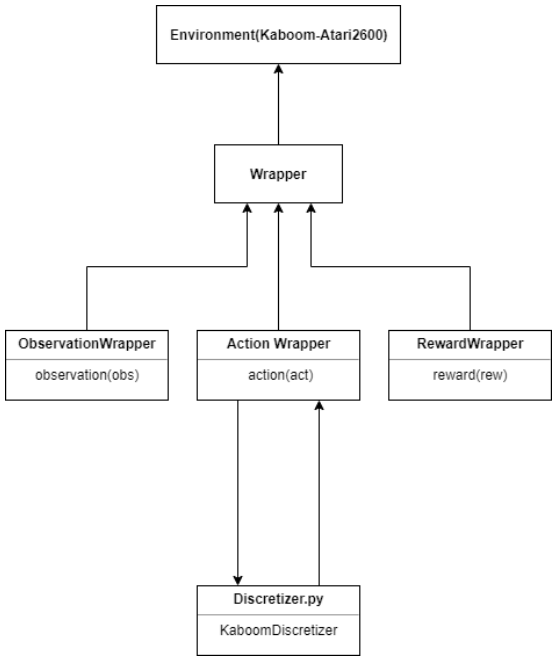
ที่ 3.4

นำเข้าไฟล์เกมสำเร็จ ทำให้สามารถสร้างสภาพแวดล้อมของเกมได้โดยใช้ฟังก์ชัน retro.make(“ชื่อเกม-ชื่อแพลตฟอร์ม”) เพื่อนำไปใช้ต่อไป โดยเกมที่นำไปใช้จะมีชื่อเกมว่า Kaboom จากเครื่อง Atari 2600 โดยชื่อสภาพแวดล้อมที่อยู่ใน Gym-Retro จะมีชื่อว่า “Kaboom-Atari2600” โดยมี Observation Space เป็น Box(210, 160, 3) ซึ่งคือเป็นภาพขนาด 210x160 พิกเซล และมีของภาพเป็นสามสีเป็นสีประเภท RGB และ Action Space เป็น MultiBinary(8) ซึ่งก็คือการควบคุมพื้นฐานของเกม Atari2600

โดยที่สภาพแวดล้อมของเกม Kaboom จะต้องใช้ไฟล์เกมที่มีชื่อว่า “Kaboom! (Paddle) (CCE).bin”

3.3 การคัดกรองการกระทำให้เหมาะสมกับเกม

ในการสร้างสภาพแวดล้อมของ Gym-Retro ในแต่ละแพลตฟอร์มของเกมคลาสิคที่นำมาใช้ การกระทำที่ได้มาจะเปลี่ยนแปลงไปตามรูปแบบของแพลตฟอร์มที่อ้างอิงมาจากตัวควบคุมจากโปรแกรมการจำลองเกมที่มาจากตัวควบคุมจริง โดยเกมที่อยู่ในเครื่อง Atari2600 ที่จำลองการเล่นด้วยโปรแกรม Stella นั้นมีทั้งหมด 8 การกระทำ ได้แก่ ขึ้น, ลง, ซ้าย, ขวา, Button, Select และ null จึงได้ทำการเลือกการกระทำที่ต้องการที่เหมาะสมกับเกมที่เลือกนำมาใช้ โดยสภาพที่มีอยู่จึงทำการเลือกใช้การกระทำทั้งหมด 3 แบบ ได้แก่ ซ้าย, ขวา และ Button ในการเล่นเกม Kaboom ที่นำมาใช้สร้างสภาพแวดล้อม[11]

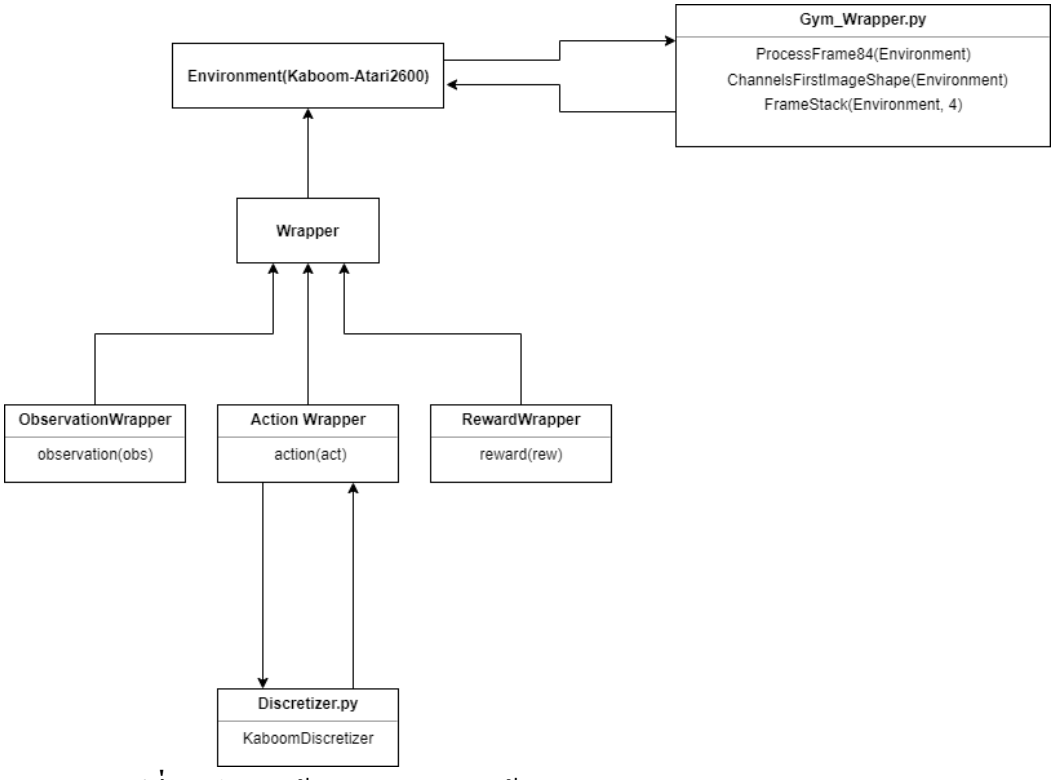


รูปที่ 3.2 โครงสร้างของ ActionWrapper และ Discretizer.py

3.4 ปรับขนาดภาพสำหรับการเป็นค่านำเข้าของโครงข่ายคอนโวลูชัน

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

การแปลงค่าตามทฤษฎีที่เกี่ยวข้องจะลดขนาดของแต่ละภาพขนาด 84x84 พิกเซล จำนวน 4 เฟรมมาต่อกันเพื่อให้ได้ค่ารับเข้าสำหรับโครงข่ายคอนโวลูชั่น โดยที่เฟรมแรกคือ เฟรมภาพปัจจุบันและอีกสามเฟรมคือเฟรมก่อนเฟรมปัจจุบันตามลำดับ [12] และจะได้โครงสร้างที่พร้อมนำไปใช้กับโครงข่ายคอนโวลูชันดังตารางที่ 3.3



รูปที่ 3.3 โครงสร้างของสภาพแวดล้อมและ Gym_Wrapper.py

ตารางที่ 3.3 โครงสร้างของสภาพแวดล้อมหลังจากการปรับ Observation สำหรับโครงข่ายคอนโวลูชัน

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

โดยที่การเตรียมค่าเพื่อที่จะให้กับโครงข่ายแบบคอนโวลูชัน จะมีไฟล์ที่ชื่อว่า Gym_Wrappers.py [13] 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 และอัลกอริทึมที่นำมาใช้นั้นมีชื่อว่า Double Deep Q Network [13][14] ซึ่งเป็นอัลกอริทึมที่พัฒนามาจาก Deep Q Network เนื่องจากเมื่อมีการทำมากขึ้น อัตราการค้นหาวิธีใหม่ของเอเจนต์จะน้อยลง และจะเลือกใช้วิธีที่ดีที่สุดของค่า Q-Value ที่จัดเก็บไว้ ซึ่งอาจจะไม่ใช่วิธีที่ดีที่สุด

โดยเมื่อเอเจนต์เริ่มทำเข้าไปอยู่ในสภาพแวดล้อมนั้น ตัวเอเจนต์จะไม่ทราบว่าจะมีอะไรเกินขึ้นในพื้นที่ จึงทำการคำนวณค่าประมาณออกมา โดยถ้าใน Deep Q Network ในการเลือกการกระทำจะอ้างอิงตามกฎของ Policy ซึ่งก็คือ จะเลือกค่า Q-Value ที่มากที่สุดในสถานะนั้น ๆ โดยในบ้างครั้งบางการกระทำอาจจะเป็นการกระทำที่ดีที่สุดกว่าที่เลือกไว้ แต่อาจจะมีค่า Q-Value ที่มีค่าน้อยกว่า หรือต่างกันเพียงเล็กน้อย แต่ว่าค่า Q-Value ที่คำนวณมีค่าที่มากที่สุดจึงทำให้เลือกการกระทำที่มีค่า Q-Value มากที่สุดแต่อาจจะไม่ใช่การกระทำที่ดีที่สุดของสถานะนั้น

Double Deep Q Network จะทำการแบ่งโครงข่ายเป็นสองส่วน ส่วนแรกเป็นส่วนที่ใช้สำหรับเลือกการกระทำโดยการเลือกการกระทำที่มีค่า Q-value ที่มากที่สุด และอีกหนึ่งส่วนเป็นส่วนของการคำนวณแวลูฟังก์ชันจากการกระทำที่ได้เลือกไว้และนำมาคำนวณภายในโครงข่ายมารวมกันโดยนำค่าน้ำหนักของโครงข่ายของการกระทำ ไปยังโครงข่ายที่ใช้สำหรับการคำนวณ Q-Value และนำไปทำตาม Policy เพื่อเลือกการกระทำเพื่อนำไปใช้ต่อไป

$$Q_{qnet}(s_t, a_t) = R_{t+1} + \gamma Q_{net}(s_{t+1}, a)$$

บทที่ 4
ผลการทดลองเบื้องต้น

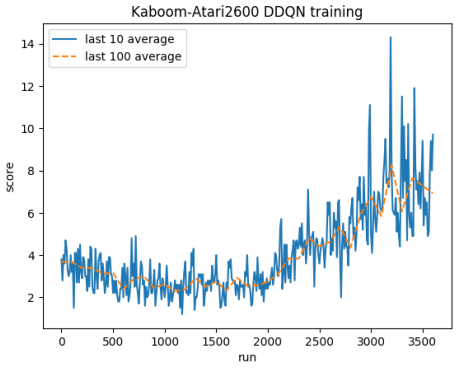
จากที่ผู้จัดทำได้ไปศึกษา ค้นคว้า และลองทำการทดลองมา โดยทำการทดลองโดยใช้เกม Kaboom ของเครื่องเกง Atari 2600 เป็นเครื่องเกมสมัยก่อน รางวัลที่ให้เอเจนต์มีด้วยกัน 3 รูปแบบ คือ -1 และ 1 โดยที่เมื่อเวลาที่สามารถรับระเบิดได้นั้น ก็จะได้รับรางวัลเป็น 1 คะแนน แต่ถ้าหากว่ารับระเบิดไม่ได้นั้นรางวัลที่ได้ก็จะเป็น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

ในการทดลองนี้ได้ใช้อัลกอริทึม DDQN (Double Deep Q Network) โดยมี Batch size ขนาด 32 โดยที่เราจะให้มีการฝึกสอนรวมกับการสุ่มการกระทำ เพื่อที่จะนำข้อมูลจากที่สุ่มการกระทำไปทำการฝึกสอนโดยกำหนดไว้ทั้งหมดคือ 5,000,000 โดยที่จะแบ่งเป็นเป็นการสุ่มการกระทำ 0.1% หรือก็คือ 50,000 การกระทำ

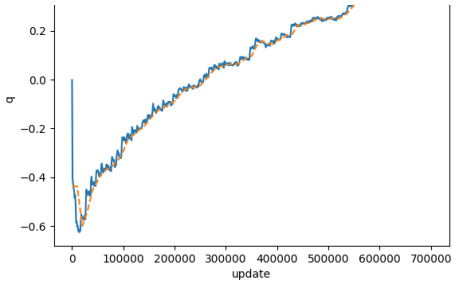
ในการทดลองนี้ได้กำหนดเกม Kaboom จากเครื่อง Atari 2600 มาทำการทดสอบ โดยผู้จัดทำต้องทำการสร้างสภาพแวดล้อมให้กับเอเจนต์เพื่อที่จะเอเจนต์สามารถทำการเล่นเกมได้ หลังจากที่สร้างสภาพแวดล้อมเสร็จ ระบบจะนำสภาพแวดล้อมที่สร้างไปให้เอเจนต์ทำการเรียนรู้ผ่านโครงสร้างของการเรียนรู้แบบเสริมกำลังที่มีโครงข่ายประสาทแบบคอนโวลูชัน โดยในการกระทำชุดแรกจะเป็นการสุ่มการกระทำจำนวนตามที่กำหนด ก็จะนำข้อมูลที่ได้ไปใช้ในการฝึกสอน หลังจากทำการเรียนรู้สำเร็จจะแสดงข้อมูลออกมาในรูปแบบกราฟ และ โมเดลสำหรับการนำไปทดสอบต่อไป

4.1 การฝึกสอน

ในการเรียนรู้แบบเสริมกำลัง จะประเมินความถูกต้องของโมเดลที่นำมาทำการฝึกสอนด้วยวิธีการให้ทดสอบกับตัวเอเจนต์ โดยคะแนนที่เรานำมาทำการสร้างกราฟนั้นเราจะใช้คะแนนเฉลี่ยทุก ๆ 10 รอบของการเล่นเกม ถ้าหากว่าคะแนนที่แสดงออกมาดีขึ้น แสดงว่าโมเดลที่นำมาทำการฝึกสอนให้กับเอเจนต์นั้นมีประสิทธิภาพ โดยปกติแล้วคนจะเล่นคะแนนเฉลี่ยประมาณ 10 คะแนน



รูปที่ 4.1 กราฟแสดงคะแนนที่เอเจนต์ทำการฝึกสอน



รูปที่ 4.2 กราฟแสดงค่า Q-Value

รูปภาพที่ 4.2 เป็นกราฟแสดงให้เห็นถึง ค่า Value function, ค่าความแม่นยำ, ค่าการสูญเสีย ตามลำดับ โดยที่ Q-Value เป็นค่าที่แสดงถึงว่าการกระทำที่เลือกมาภายใต้สถานะหนึ่งทำให้ไปถึงเป้าหมายได้ดีขึ้นเพียงใด

4.2 ประเมินผลการทดลองที่เกิดขึ้น

จากผลการทดลองที่เกิดขึ้นข้างต้นได้ประสบปัญหาในการฝึกสอนพอถึงจุดช่วงหนึ่งเครื่องคอมพิวเตอร์ที่ใช้ในการฝึกสอนเกิดอาการหยุดการตอบสนอง ทำให้ไม่สามารถทดลองฝึกสอนให้กับปัญญาประดิษฐ์ได้ตามที่กำหนด การกระทำที่เกิดขึ้นก่อนที่เครื่องคอมพิวเตอร์จะเกิดอาการไม่ตอบสนองคืออยู่ในช่วงการกระทำที่ 2,000,000 การกระทำ ทำให้ไม่สามารถระบุได้ชัดเจนว่าโมเดลที่ถูกฝึกสอนให้ผลลัพธ์ที่ดีมากน้อยเพียงใด

เนื่องจากเกิดข้อผิดพลาดจึงทำให้เราไม่ได้นำโมเดลที่ได้ไปทำการทดสอบดูว่าดีมากน้อยแค่ไหน เพราะเป็นโมเดลที่ไม่สมบูรณ์

บทที่ 5
บทสรุป

5.1 สรุปผลการดำเนินงาน

จากการได้ทำการศึกษาเรื่องการเรียนรู้แบบเสริมกำลัง ทำให้เราทราบว่าการเรียนรู้แบบเสริมกำลังมีโครงสร้าง และมีหลักการทำงานเป็นอย่างไร ผู้จัดทำได้ใช้เวลา 2 อาทิตย์ในการทำความเข้าใจกับการเรียนรู้แบบเสริมกำลังคร่าว ๆ ทำการค้นคว้าว่าไลบรารีที่จำเป็นต้องใช้ในการทำการเรียนรู้แบบเสริมกำลัง หลังจากนั้นได้ทดลองสร้างสภาพแวดล้อมให้กับเกมที่ต้องการจะนำมาใช้กับการเรียนรู้แบบเสริมกำลัง ต่อมาได้ทำการศึกษาวิธีที่จะใช้ในการฝึกสอนให้กับปัญญาประดิษฐ์ ว่ามีวิธีใดบ้างที่จะได้ประสิทธิภาพบ้าง เราจึงได้พบว่ามีวิธีการใช้ DDQN อัลกอริทึม ที่ช่วยในการฝึกสอนให้กับปัญญาประดิษฐ์นั้นได้ผลลัพธ์ที่ดีกับเกมรูปแบบอื่น ทำให้มีความสนใจกับอัลกอริทึมนี้และนำมาทดลองใช้

5.2 ปัญหาและอุปสรรค

เนื่องจากปัญหาคอมพิวเตอร์ไม่มีการตอบสนองในระหว่างการฝึกสอนให้กับปัญญาประดิษฐ์ ทำให้ไม่สามารถทำการฝึกสอนต่อ และการเรียนรู้แบบเสริมกำลังเป็นเรื่องที่มีผู้ศึกษาไม่มากนักทำให้การสืบค้นข้อมูลในช่วงแรกเป็นไปค่อนข้างลำบาก และวิทยานิพนธ์ที่สืบค้นส่วนใหญ่เป็นเนื้อที่ใหญ่กว่าโครงงานของเราเป็นอย่างมาก

5.3 แผนงานสำหรับการศึกษาต่อ

สิ่งที่สนใจจะศึกษากันต่อไปคือ ผู้จัดทำจะนำอัลกอริทึมอื่น ๆ ที่ใช้ในการฝึกสอนให้กับปัญญาประดิษฐ์มาเปรียบเทียบเพื่อค้นหาอัลกอริทึมที่มีความเหมาะสมกับเกมนี้มากที่สุด และทดลองนำโมเดลที่ได้จากการทดสอบไปใช้กับเกมจริง เพื่อทดสอบว่าโมเดลที่ได้รับการทดสอบมีปัญหากับการนำเกมจริงมาใช้ทดสอบเล่นหรือไม่

บรรณานุกรม

[1] Vinyals, Oriol, et al. "Starcraft ii: A new challenge for reinforcement learning." arXiv preprint arXiv:1708.04782 (2017).

[2] David Silver (2015), “Introduction of reinforcement learning” [PowerPoint Presentation] Advanced Topics 2015 (COMPM050/COMPGI13) Reinforcement Learning

[3] David Silver (2015), “Markov Decision Process” [PowerPoint Presentation] Advanced Topics 2015 (COMPM050/COMPGI13) Reinforcement Learning

[4] Brockman, Greg, et al. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).

[5] Nichol, Alex, et al. "Gotta learn fast: A new benchmark for generalization in rl." arXiv preprint arXiv:1804.03720 (2018).

[6] Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." Nature 518.7540 (2015): 529.

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

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

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

[10] https://towardsdatascience.com/atari-reinforcement-learning-in-depth-part-1-ddqn-ceaa762a546f เข้าถึงข้อมูลเมื่อ 28/11/2019

[11] https://github.com/openai/retro-baselines/blob/master/agents/sonic_util.py เข้าถึงข้อมูลเมื่อ 28/11/2019

[12] https://www.freecodecamp.org/news/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8/ เข้าถึงข้อมูลเมื่อ 28/11/2019

[13] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." Thirtieth AAAI conference on artificial intelligence. 2016.

[14] Hasselt, Hado V. "Double Q-learning." Advances in Neural Information Processing Systems. 2010.