การฝึกพฤติกรรมของตัวละครที่ผู้เล่นไม่ได้ควบคุมในเกมโดยวิธีการเรียนแบบเสริมกำลัง

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

โดย

สุรเชษฐ์ ใหญ่ธรรมสาร

อัครพล อัครสูริย์

อาจารย์ที่ปรึกษา

ดร. สามารถ หมุดและ

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

(.....)

.....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 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 และนำวิธีการเรียนรู้พื้นฐานของการเรียนรู้แบบเสริมกำลังมาทำการสอนให้คอมพิวเตอร์เรียนรู้และสามารถเล่นได้เอง ผู้จัดต้องการที่จะทราบว่าวิธีการเรียนรู้แบบเสริมกำลังแบบใดเหมาะกับเกมที่นำมาใช้เป็นสภาพแวดล้อมในการเล่น และนำมาเปรียบกับการเล่นกับมนุษย์

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 and compared with human plays

กิตติกรรมประกาศ

ปริญญานิพธ์นี้สำเร็จลุล่วงได้ด้วยความกรุณาจากด็อกเตอร์ สามารถ หมุดและ และด็อกเตอร์ 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 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 ที่ผ่านการพัฒนาโดยอัลกอริทึมต่าง ๆ ของการเรียนรู้แบบเสริมกำลัง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

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

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

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

2.1.1.1.1 Policy

เป็นกฎเกณฑ์ของวิธีการที่จะให้ไปถึงเป้าหมายที่ต้องการเมื่ออยู่ในสถานะที่ต่างกันออกไปเพื่อให้วิธีที่ดีที่สุดในการทำเป้าหมาย

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)

เป็นสถานะของสภาพแวดล้อมในช่วงเวลาต่าง ๆที่ปัญญาประดิษฐ์สามารถรับรู้เพื่อตัดสินใจเลือกการกระทำในแต่ละช่วงเวลา

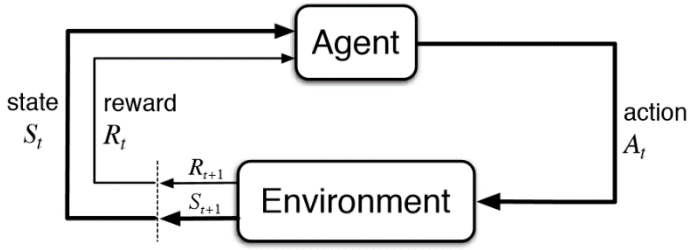
2.1.1.4 การกระทำ (Action)

เอเจนต์จะเลือกการกระทำที่ทำการตัดสินใจจากสถานะก่อนหน้าเข้าไปยังสภาพแวดล้อมเพื่อแสดงสถานะถัดไปและรางวัลที่ได้จากการเลือกการกระทำ

2.1.1.5 รางวัล (Reward)

เป็นรางวัลที่ได้จากการกระทำเพื่อแสดงว่าการกระทำที่เลือกไปดีต่อกับสภาพแวดล้อมและเป้าหมายของสภาพแวดล้อมได้ดีเพียงใด

โดยโครงสร้างของการเรียนรู้แบบเสริมกำลังจะมีวงจรเป็นการวนซ้ำของการกระทำของเอเจนต์ไปยังสภาพแวดล้อมและผลที่เกิดขึ้นและรางวัลที่ได้จากการกระทำไปยังเอเจนต์ ดังรูปที่ 2.1



รูป 2.1 โครงสร้างการทำงานของการเรียนรู้แบบเสริมกำลัง

ซึ่งการแก้ปัญหาที่ให้การเรียนรู้แบบเสริมกำลังส่วนใหญ่มักจะแก้ไขด้วยหลักการที่ชื่อว่ากระบวนการตัดสินใจของมาร์คอฟ[3] (Markov Decision Process) ซึ่งมีคุณสมบัติดังนี้

สถานะใด ๆ สถานะหนึ่งจะเป็นสถานะมาร์คอฟได้ก็ต่อเมื่อ สถานะปัจจุบันนั้นส่งผลถึงสถานะในอนาคต เท่ากับสถานะปัจจุบันและสถานะก่อนหน้าส่งผลกับสถานะในอนาคต ซึ่งทำให้ไม่ต้องสนใจสถานะในอดีตอีกต่อไป

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

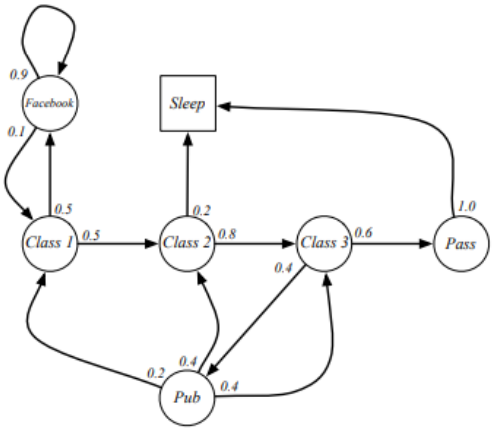
State Transition Matrix[3] เป็นเมทริกซ์ของความน่าจะเป็นของสถานะที่อยู่ไปยังสถานะต่อไปซึ่งแต่ละจุดจะมีความน่าจะเป็นดังสมการ

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

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)[3] 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[13]

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

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

แวลูฟังก์ชัน (Value Function) เป็นผลรวมของรางวัล ณ สถานะหนึ่งเพื่อแสดงว่าการกระทำที่เลือกมานั้นส่งผลดีหรือพาไปยังเป้าหมายได้ดีเพียงใดซึ่ง Value Function 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 คืออะไร?

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

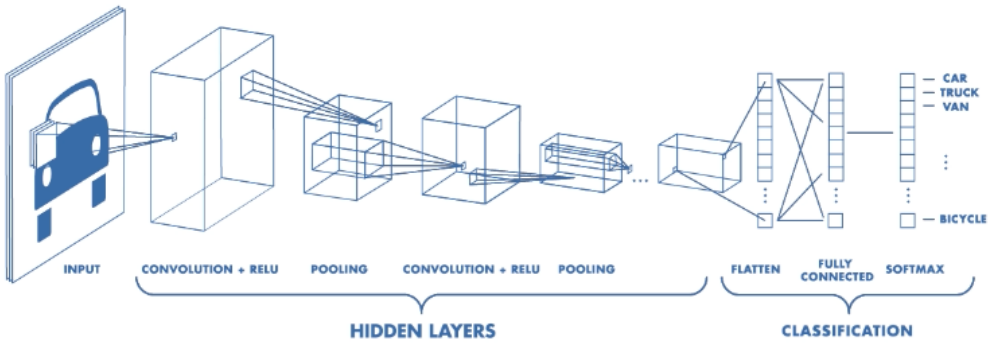
Gym-retro[5] คือไลบรารีโอเพนซอร์ซที่ทาง OpenAI สร้างขึ้นโดยมีรากฐานเดียวกับ Gym คือ ให้ผู้ที่สนใจพัฒนาในการเรียนรู้แบบเสริมกำลัง แต่ว่ามีความแตกต่างที่สภาพแวดล้อมของ Gym-retro จะเป็นเกมคลาสสิคที่เกิดในยุคแรกของอุตสาหกรรมวีดิโอเกม(ในช่วงประมาณ ค.ศ.1970 - ค.ศ.1995) เช่น Space Invader, Super Mario Bros,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



รูป 2.4 โครงสร้างของโครงข่ายประสาทแบบคอนโวลูชัน

องค์ประกอบของโครงข่ายคอนโวลูชันมีดังนี้

2.1.4.1 Convolutional Layer

เป็นชั้นที่ทำการสแกนค่ารับเข้าซึ่งเป็นรูปภาพ เพื่อแยกองค์ประกอบของรูป เช่น สี รูปทรง ขอบของภาพ

2.1.4.2 Pooling Layer

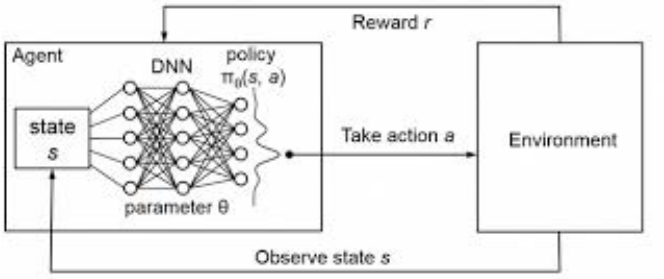
เป็นการลดขนาดของข้อมูลมีขนาดเล็กลงโดยที่รายละเอียดยังคงเดิม ซึ่งทั้ง Convolutional Layer และ Pooling Layer จะทำงานคู่กันซึ่งจะทำงานหลายครั้งเพื่อจำแนกได้ครบทุกรูปแบบ

2.1.4.3 Fully-Connected Layer

เป็นชั้นที่มีค่านำเข้าเป็นข้อมูลจากการกระทำของข้อมูลจากชั้นก่อนหน้าสำหรับนำมาคำนวณเพื่อจำแนกประเภทจากข้อมูลที่ได้มา

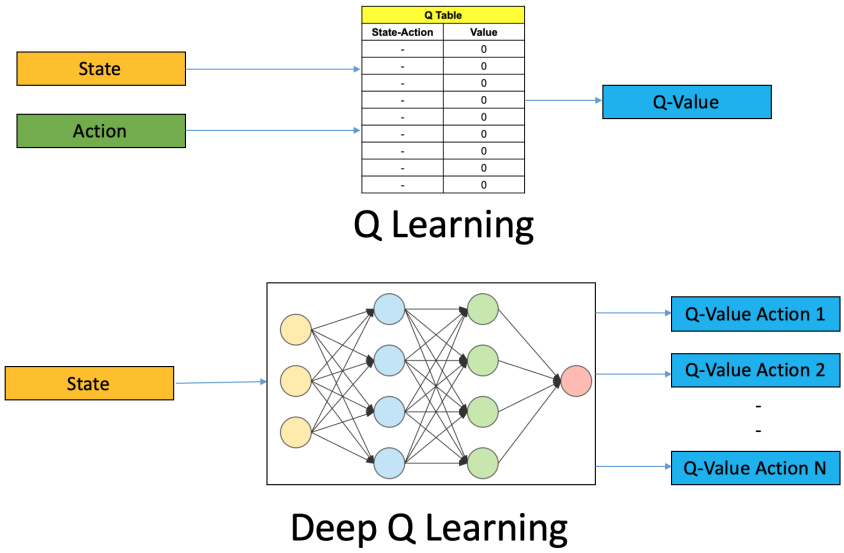
2.1.3 การเรียนรู้แบบเสริมกำลังเชิงลึก (Deep Reinforcement Learning)

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 (2556)ได้ทำการสร้างโมเดลสำหรับการเรียนรู้ โดยใช้โครงข่ายคอนโวลูชัน และใช้รูปแบบการเรียนรู้ Q-Learning ของการเรียนรู้แบบเสริมกำลัง ซึ่งเรียกว่า Deep Q Networkโดยใช้รูปภาพสำหรับค่านำเข้า 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 จึงเป็นค่าสำคัญมากที่ส่งผลต่อการทำเป้าหมายให้สำเร็จได้และนำโครงข่ายคอนโวลูชันมาใช้รวมกันเพื่อทำการฝึกสอน

2.2.2 การเรียนรู้แบบเสริมกำลังโดยใช้วิธีอะซิงโครนัส (Asynchronous Methods for Deep Reinforcement Learning)

กลุ่ม Deepmind (2559) ได้ทำการสร้างโมเดลแบบใหม่ที่สามารถทำการเรียนรู้แบบคู่ขนานชื่อว่า A3C (Asynchronous Actor-Critic Agents) โดยแบ่งแต่ละกลุ่มโดยใช้อัลกอริทึมของการเรียนรู้แบบเสริมกำลังที่ต่างกัน 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 เป็นฐานของอัลกอริทึมใหม่ เพื่อลดค่าที่เกินออกมามากผิดปกติและทำการชะลอการอัพเดท 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 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 [10]

2.2.5 การเรียนรู้แบบเสริมกำลังโดยใช้วิธีการ Double Deep Q Network

เกรก เซอร์มา (Greg Surma) ได้ทำสร้างสร้างการเรียนรู้แบบเสริมกำลังโดยใช้วิธีการ Double Deep Q Network เพื่อพิสูจน์ว่าเอเจนต์ที่ทำงานโดยอัลกอริทึมนั้นสามารถในการแก้ปัญหาในสภาพแวดล้อมแบบต่าง ๆ ได้หรือไม่[11]

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

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

บทที่ 3
วิธีการดำเนินงาน

3.1 โปรแกรมหรือซอฟต์แวร์ที่ใช้ในการพัฒนา

3.1.1 ภาษาไพทอน

สำหรับการเขียนโครงสร้างของโครงงาน ซึ่งประกอบไปด้วยไลบรารี ดังนี้

3.1.1.1 Gym

เป็นไลบรารีสำหรับการพัฒนาและเปรียบเทียบอัลกอริทึมของการเรียนรู้แบบเสริมกำลัง โดยเป็นส่วนหลักที่นำมาเป็นโครงสร้างของการพัฒนาการเรียนรู้แบบเสริมกำลัง

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 โดยในโครงงานจะใช้ไลบรารีในการสร้างสภาพแวดล้อม และการเรียนรู้แบบเสริมกำลังจะใช้จากไลบรารี Gym เป็นหลัก

3.1.1.3 Numpy

เป็นไลบรารีที่ใช้สร้างสูตรการคำนวณที่เกี่ยวข้องกับคณิตศาสตร์ ภายในโครงงานนี้จะทำการเก็บค่าของการเรียนรู้แบบเสริมกำลัง และการจัดเก็บและดัดแปลงข้อมูลให้อยู่ในรูปแบบของเมทริกซ์ และแปลงรูปภาพของสภาพแวดล้อมเป็นอาเรย์ขนาดใหม่ที่ใช้สำหรับการนำไปประมวลผล

3.1.1.4 Matplotlib

เป็นไลบรารีสำหรับการสร้างแผนภูมิสำหรับการวิเคราะห์ข้อมูล นำมาใช้สำหรับการแสดงผลการทดลองออกมาทางรูปแบบของแผนภูมิของโครงงานในหัวข้อต่าง ๆ เพื่อนำมาสรุปผลการทดลอง

3.1.1.5 Keras

เป็นไลบรารี Deep learning ที่นำมาใช้ร่วมกับการทำงานของ การเรียนรู้แบบเสริมกำลัง ให้เป็นการทำงานแบบเชิงลึกเพื่อเพิ่มประสิทธิภาพการทำงานให้สูงขึ้นภายในโครงงานได้นำมาใช้สร้างโครงข่ายประสาทแบบคอนโวลูชันเพื่อนำมาประมวลผลของเอเจนต์

3.1.1.6 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

3.2 ขั้นตอนการดำเนินงาน

3.2.1 สร้างสภาพแวดล้อมสำหรับการนำเอเจนต์อยู่ในพื้นที่ที่กำหนด

สภาพแวดล้อมที่ผู้จัดทำใช้ไลบรารี Gym-retro ในการนำสภาพแวดล้อมที่เป็นเกมคลาสสิก ซึ่งผู้ใช้ต้องนำไฟล์เกมที่ไลบรารีต้องการ เพื่อสามารถสร้างสภาพแวดล้อมที่ไลบรารีสนับสนุน โดยเกมที่ผู้จัดทำใช้เป็นสภาพแวดล้อมคือเกม Kaboom จากเครื่อง อาตาริ 2600 ดังรูปที่ 3.1

รูปที่ 3.1 ภาพของเกม Kaboom จากเครื่อง Atari2600

ต้องทำการนำเข้าไฟล์เกมของ Kaboom โดยใช้ชื่อไฟล์ว่า “Kaboom! (Paddle) (CCE).bin” เพื่อสามารถสร้างสภาพแวดล้อมได้ เพราะไลบรารี gym-retro[5] ต้องการไฟล์เกมที่ถูกต้องในการสร้างสภาพแวดล้อม

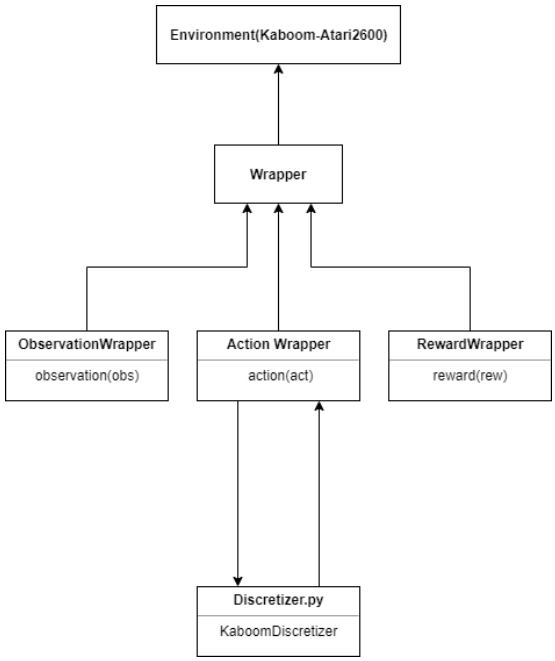
ทำการสร้างสภาพแวดล้อมโดยใช้ฟังก์ชั่น retro.make() จากไลบรารี gym-retro เพื่อสร้างสภาพแวดล้อมและมีลักษณะโครงสร้าง ดังตารางที่ 3.1

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 ต้องการเลือกบางการกระทำที่เหมาะสมกับสภาพแวดล้อมที่กำหนด โดยเกม Kaboom ใช้ปุ่มทั้งหมด 3 ปุ่ม ได้แก่ ซ้าย, ขวา, Button จึงได้สร้างไฟล์ Discretizer.py[12] สำหรับการคัดเลือกการกระทำที่ต้องการ



รูปที่ 3.2 โครงสร้างของ ActionWrapper และ Discretizer.py

ซึ่งได้ทำการเปลี่ยนการกระทำให้เหลือเพียงสามแบบซึ่งเปลี่ยนการกระทำในสภาพแวดล้อมดังตารางที่ 3.2

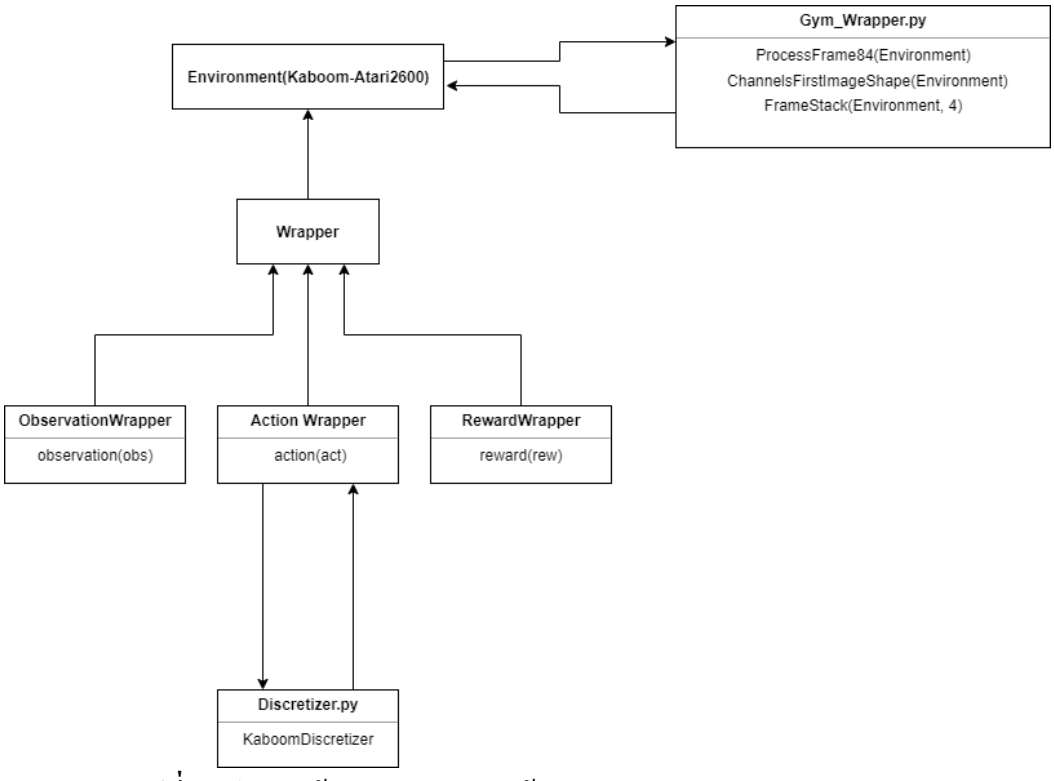
ชื่อสภาพแวดล้อม	Kaboom-Atari2600
Observation Space	Box (210, 160, 3)
Action Space	Discrete (8)

ตารางที่ 3.2 โครงสร้างของสภาพแวดล้อมหลังจากทำการแยกการกระทำที่ต้องการ

3.2.3 ปรับขนาดภาพสำหรับการเป็นค่านำเข้าของโครงข่ายคอนโวลูชัน

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

การแปลงค่าตามทฤษฎีที่เกี่ยวข้องจะลดขนาดของแต่ละภาพขนาด 84x84 พิกเซล จำนวน 4 เฟรมมาต่อกันเพื่อให้ได้ค่ารับเข้าสำหรับโครงข่ายคอนโวลูชั่น โดยที่เฟรมแรกคือ เฟรมภาพปัจจุบันและอีกสามเฟรมคือเฟรมก่อนเฟรมปัจจุบันตามลำดับ[13]



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[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

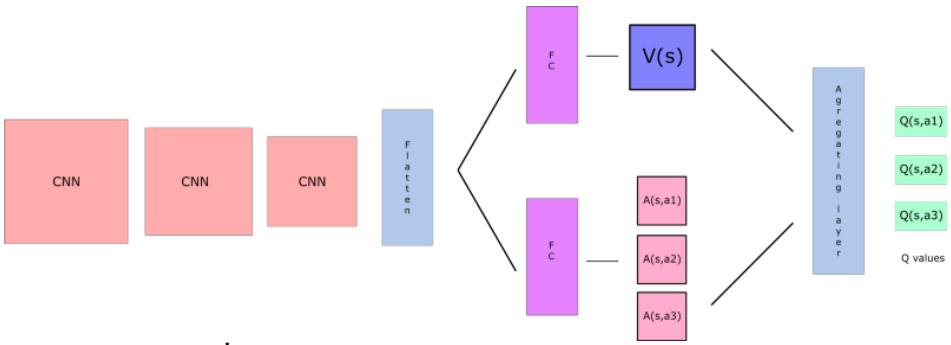
คลาส StackFrame เป็นคลาสที่ทำการซ้อนภาพจำนวน 4 ภาพเพื่อที่ทำการนำมาเป็นค่านำเข้าของโครงข่ายคอนโวลูชัน โดยมีคลาส LazyFrame ที่ช่วยให้การจัดเก็บเกิดขึ้นภายในครั้งเดียวเพื่อประหยัดทรัพยากร

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

เมื่อผ่านคลาสทั้งหมดที่กล่าวมาจะได้สภาพแวดล้อมที่นำไปใช้กับโครงสร้างคอนโวลูชันต่อไป

อัลกอริทึมที่นำมาใช้นั้นมีชื่อว่า Double Deep Q Network[14][15] ซึ่งเป็นอัลกอริทึมที่พัฒนามาจาก Deep Q Network เนื่องจากเมื่อมีการทำมากขึ้น อัตราการค้นหาวิธีใหม่ของเอเจนต์จะน้อยลง และจะเลือกใช้วิธีที่ดีที่สุดของค่า Q-Value ที่จัดเก็บไว้ ซึ่งอาจจะไม่ใช่วิธีที่ดีที่สุด

Double Deep Q Network จะทำการแบ่งโครงข่ายเป็นสองส่วน ส่วนแรกเป็นส่วนที่ใช้สำหรับเลือกการกระทำ และอีกหนึ่งส่วนเป็นส่วนของการคำนวณแวลูฟังก์ชันจากการกระทำที่ได้เลือกไว้และนำทั้งสองโครงข่ายมารวมกันโดยนำค่าน้ำหนักของโครงข่ายของการกระทำ ไปยังโครงข่ายที่ใช้สำหรับการคำนวณ Q-Value



รูปที่ 3.4 โครงสร้างของโครงข่าย Double Deep Q Network

บทที่ 4
ผลการทดลองเบื้องต้น

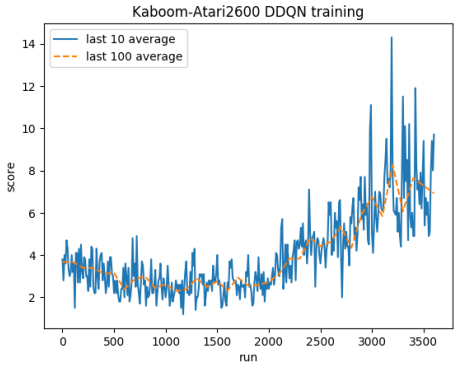
จากที่ผู้จัดทำได้ไปศึกษา ค้นคว้า และลองทำการทดลองมา โดยทำการทดลองโดยใช้เกม Kaboom ของเครื่องเกง Atari 2600 เป็นเครื่องเกมสมัยก่อน รางวัลที่ให้เอเจนต์มีด้วยกัน 3 รูปแบบ คือ -1 และ 1 โดยที่เมื่อเวลาที่สามารถรับระเบิดได้นั้น ก็จะได้รับรางวัลเป็น 1 คะแนน แต่ถ้าหากว่ารับระเบิดไม่ได้นั้นรางวัลที่ได้ก็จะเป็น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

ในการทดลองนี้ได้ใช้อัลกอริทึม DDQN (Double Deep Q Network)[14][15] โดยมี Batch size ขนาด 32 โดยที่เราจะให้มีการฝึกสอนรวมกับการสุ่มการกระทำ เพื่อที่จะนำข้อมูลจากที่สุ่มการกระทำไปทำการฝึกสอนโดยกำหนดไว้ทั้งหมดคือ 5,000,000 โดยที่จะแบ่งเป็นเป็นการสุ่มการกระทำ 0.1% หรือก็คือ 50,000 การกระทำ

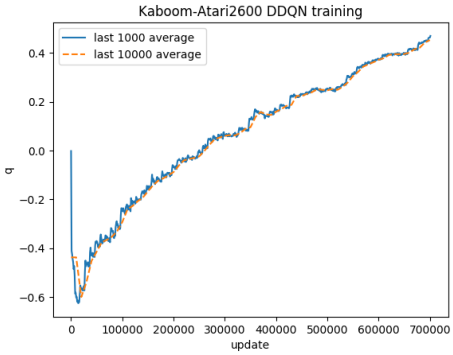
ในการทดลองนี้ได้กำหนดเกม Kaboom จากเครื่อง Atari 2600 มาทำการทดสอบ โดยผู้จัดทำต้องทำการสร้างสภาพแวดล้อมให้กับเอเจนต์เพื่อที่จะเอเจนต์สามารถทำการเล่นเกมได้ หลังจากที่สร้างสภาพแวดล้อมเสร็จ ระบบจะนำสภาพแวดล้อมที่สร้างไปให้เอเจนต์ทำการเรียนรู้ผ่านโครงสร้างของการเรียนรู้แบบเสริมกำลังที่มีโครงข่ายประสาทแบบคอนโวลูชัน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

4.1 การฝึกสอน

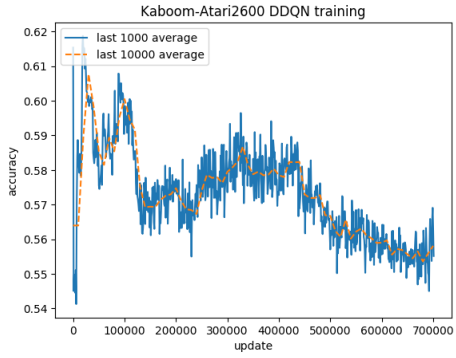
ในการเรียนรู้แบบเสริมกำลัง จะประเมินความถูกต้องของโมเดลที่นำมาทำการฝึกสอน ด้วยวิธีการให้ทดสอบกับตัวเอเจนต์ โดยคะแนนที่เรานำมาทำการสร้างกราฟนั้นเราจะใช้คะแนนเฉลี่ยทุก ๆ 10 รอบของการเล่นเกม ถ้าหากว่าคะแนนที่แสดงออกมาดีขึ้น แสดงว่าโมเดลที่นำมาทำการฝึกสอนให้กับเอเจนต์นั้นมีประสิทธิภาพ โดยปกติแล้วคนจะเล่นคะแนนเฉลี่ยประมาณ 10 คะแนน



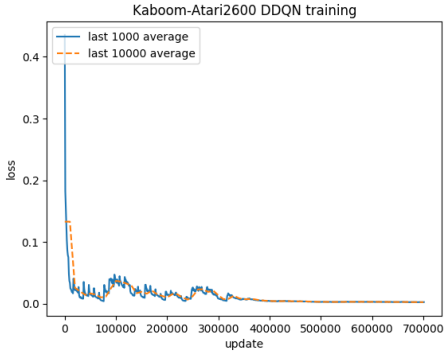
รูปที่ 4.1 กราฟแสดงคะแนนที่เอเจนต์ทำการฝึกสอน



รูปที่ 4.2 กราฟแสดงค่า Q-Value



รูปที่ 4.3 กราฟแสดงค่า Accuracy



รูปที่ 4.4 กราฟแสดงค่า Loss

รูปภาพที่ 4.2, 4.3 และ 4.4 เป็นกราฟแสดงให้เห็นถึง ค่า Value function, ค่าความแม่นยำ, ค่าการสูญเสีย ตามลำดับ โดยที่ Q-Value เป็นค่าที่ .....???..... ถัดมาคือ Accurency บ่งบอกถึงความแม่นยำของข้อมูลยิ่งกราฟยิ่งมีขนาดสูงเท่าใดจะหมายถึงข้อมูล และสุดท้ายคือ Loss

4.2 ประเมินผลการทดลองที่เกิดขึ้น

จากผลการทดลองที่เกิดขึ้นข้างต้นได้ประสบปัญหาในการฝึกสอนพอถึงจุดช่วงหนึ่งเครื่องคอมพิวเตอร์ที่ใช้ในการฝึกสอนเกิดอาการหยุดการตอบสนอง ทำให้ไม่สามารถทดลองฝึกสอนให้กับปัญญาประดิษฐ์ได้ตามที่กำหนด การกระทำที่เกิดขึ้นก่อนที่เครื่องคอมพิวเตอร์จะเกิดอาการไม่ตอบสนองคืออยู่ในช่วงการกระทำที่ 2,000,000 การกระทำ ทำให้ไม่สามารถระบุได้ชัดเจนว่า โมเดลที่ถูกฝึกสอนให้ผลลัพธ์ที่ดีมากน้อยเพียงใด

เนื่องจากเกิดข้อผิดพลาดจึงทำให้เราไม่ได้นำโมเดลที่ได้ไปทำการทดสอบดูว่าดีมากน้อยแค่ไหน เพราะเป็นโมเดลที่ไม่สมบูรณ์

บทที่ 5
บทสรุป

5.1 สรุปผลการดำเนินงาน

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

สิ่งที่สนใจจะศึกษากันต่อไปคือ ผู้จัดทำจะนำอัลกอริทึมอื่น ๆ ที่ใช้ในการฝึกสอนให้กับปัญญาประดิษฐ์มาเปรียบเทียบเพื่อค้นหาอัลกอริทึมที่มีความเหมาะสมกับเกมนี้มากที่สุด และทดลองนำโมเดลที่ได้จากการทดสอบไปใช้กับเกมจริง เพื่อทดสอบว่าโมเดลที่ได้รับการทดสอบมีปัญหากับการนำเกมจริงมาใช้ทดสอบเล่นหรือไม่

5.2 ปัญหาและอุปสรรค

เนื่องจากปัญหาคอมพิวเตอร์ไม่มีการตอบสนองในระหว่างการฝึกสอนให้กับปัญญาประดิษฐ์ ทำให้ไม่สามารถทำการฝึกสอนต่อ และการเรียนรู้แบบเสริมกำลังเป็นเรื่องที่มีผู้ศึกษาไม่มากนักทำให้การสืบค้นข้อมูลในช่วงแรกเป็นไปค่อนข้างลำบาก และวิทยานิพนธ์ที่สืบค้นส่วนใหญ่เป็นเนื้อที่ใหญ่กว่าโครงงานของเราเป็นอย่างมาก

บรรณาณุกรม

[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] 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.

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

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

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

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

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