โปรแกรมประยุกต์การระบุตำแหน่งภายในอาคาร
ด้วยสัญญาณ RSSI จากเทคโนโลยี IoT

INDOOR POSITIONING APPLICATION USING RSSI
BASED ON IoT TECHNOLOGIES

โดย

นายวีรชน พรหมทะสาร

นายจิรภัทร คำนวน

นางสาวพรทิตา มีสัตย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2564

โปรแกรมประยุกต์การระบุตำแหน่งภายในอาคาร
ด้วยสัญญาณ RSSI จากเทคโนโลยี IoT

INDOOR POSITIONING APPLICATION USING RSSI
BASED ON IoT TECHNOLOGIES

โดย

นายวีรชน	พรหมทะสาร	61010985
นายจิรภัทร	คำนวน	61011152
นางสาวพรทิตา	มีสัตย์	61011402

อาจารย์ที่ปรึกษา

รศ. ดร.พิสิฐ บุญศรีเมือง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2564

ปริญญานิพนธ์ปีการศึกษา 2564

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมประยุกต์การระบุตำแหน่งภายในอาคารด้วยสัญญาณ RSSI จากเทคโนโลยี IoT
INDOOR POSITIONING APPLICATION USING RSSI BASED ON IoT TECHNOLOGIES

ผู้จัดทำ

1. นายวีรชน พรหมทะสาร 61010985
2. นายจิรภัทร คำนวน 61011152
3. นางสาวพรทิตา มีสัตย์ 61011402

..... **อาจารย์ที่ปรึกษา**

(รศ. ดร.พิสิฐ บุญศรีเมือง)

กิตติกรรมประกาศ

ปริญญานิพนธ์ “โปรแกรมประยุกต์การระบุตำแหน่งภายในอาคารด้วยสัญญาณ RSSI จากเทคโนโลยี IoT” จะไม่สามารถสำเร็จลุล่วงไปได้ หากไม่ได้รับความอนุเคราะห์อย่างดียิ่งจาก รศ.ดร.พิสิฐ บุญศรีเมือง ที่คอยให้คำแนะนำและแนวทางแก้ไขปัญหาที่เป็นประโยชน์ รวมถึงสนับสนุนเครื่องมือและอุปกรณ์ต่างๆ ที่ใช้ในปริญญานิพนธ์นี้

ขอขอบคุณรุ่นพี่และเพื่อนในภาควิชาโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้ให้คำปรึกษา แนะนำ และถ่ายทอดวิชาความรู้ให้แก่คณะผู้จัดทำ

สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา และครอบครัวที่ให้ความรัก ความห่วงใย และที่สำคัญคือสนับสนุนให้โอกาสทางด้านการศึกษาอันมีค่ายิ่งแก่คณะผู้จัดทำ หากมีข้อบกพร่องประการใด คณะผู้จัดทำยินดีน้อมรับคำติชมจากทุกท่านที่ได้เข้ามาศึกษา เพื่อเป็นประโยชน์ในการพัฒนาต่อไป

นายวีรชน พรหมทะสาร
นายจิรภัทร คำนวณ
นางสาวพรทิตา มีสัตย์
ผู้จัดทำ

โปรแกรมประยุกต์การระบุตำแหน่งภายในอาคารด้วยสัญญาณ RSSI จากเทคโนโลยี IoT

INDOOR POSITIONING APPLICATION USING RSSI BASED ON IoT TECHNOLOGIES

โดย	นายวีรชน	พรหมทะสาร	61010985
	นายจิรภัทร	คำนวณ	61011152
	นางสาวพรทิตา	มีสัตย์	61011402

อาจารย์ที่ปรึกษา รศ. ดร.พิสิฐ บุญศรีเมือง

บทคัดย่อ

เนื่องจากในปัจจุบันเทคโนโลยีระบบกำหนดตำแหน่งบนพื้นโลก (global positioning system: GPS) ถูกใช้กันอย่างแพร่หลาย แต่ยังมีข้อจำกัดคือไม่สามารถใช้ระบุตำแหน่งภายในอาคาร เนื่องจากสัญญาณดาวเทียมไม่สามารถทะลุผ่านเข้าไปในอาคารได้ ปริญญานิพนธ์นี้จึงมีวัตถุประสงค์เพื่อออกแบบระบบระบุตำแหน่งภายในอาคารด้วยค่าตัวชี้บอกความแรงของสัญญาณที่ได้รับ (Receive Signal Strength Indicator: RSSI) โดยจะนำค่า RSSI มาทำการวิเคราะห์ด้วยอัลกอริทึมการเรียนรู้เชิงลึกและกระบวนการอื่นๆ เพื่อให้สามารถระบุตำแหน่งได้อย่างแม่นยำ และพัฒนาเป็นโปรแกรมประยุกต์ที่ใช้ระบุตำแหน่งบนโทรศัพท์เคลื่อนที่ ซึ่งระบบระบุตำแหน่งภายในอาคารที่นำเสนอแบ่งออกเป็น 3 ส่วนหลักๆ คือ 1. ส่วนนำเข้าข้อมูล ซึ่งจะรับค่า RSSI และส่งต่อไปยังส่วนประมวลผล 2. ส่วนประมวลผลข้อมูล จะรับข้อมูลค่า RSSI จากส่วนนำเข้าข้อมูล เพื่อนำมาวิเคราะห์และทำนายตำแหน่งภายในอาคาร และ 3. ส่วนส่งออกข้อมูล โดยจะส่งผลการทำนายตำแหน่งที่ได้จากส่วนประมวลผลแสดงกลับไปยังโปรแกรมประยุกต์บนโทรศัพท์เคลื่อนที่

ABSTRACT

Nowadays global positioning system (GPS) technology is widely used. But there is still a limitation that cannot be used to determine the location inside the building because the satellite signal cannot penetrate the building. The objective of this thesis is to design an indoor positioning system with Receive Signal Strength Indicator (RSSI). RSSI values are analyzed by deep learning algorithms and other processes to be able to pinpoint the location accurately and developed as an application that uses location on smartphones. The proposed indoor positioning system is divided into 3 main parts: 1. Data import section; which receives the RSSI values and forwards them to the processor 2. Data processing section; it receives the RSSI data from the data input section to analyze and predict the location inside the building and 3. Data export section; the results of the prediction of the position obtained from the processor are displayed back to the smartphone application.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญตาราง

หน้า

ตารางที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปริญญานิพนธ์นี้มีแรงบันดาลใจมาจากเทคโนโลยีระบบกำหนดตำแหน่งบนพื้นโลก (global positioning system: GPS) ที่มีการใช้งานกันอย่างแพร่หลาย สามารถใช้ประโยชน์ได้มากมาย ทั้งเป็นระบบติดตามและนำทาง แต่ GPS นั้นยังมีข้อจำกัดคือ ไม่สามารถใช้ระบุตำแหน่งภายในอาคาร เนื่องจากสัญญาณดาวเทียมไม่สามารถทะลุผ่านเข้าไปในอาคารได้ และเพื่อก้าวข้ามข้อจำกัดดังกล่าว จึงเป็นเหตุจูงใจให้นำเสนอปริญญานิพนธ์นี้

โปรแกรมประยุกต์การระบุตำแหน่งภายในอาคารด้วยค่าตัวชี้บอกความแรงของสัญญาณที่ได้รับ (Receive Signal Strength Indicator: RSSI) จากเทคโนโลยีโครงข่ายของสรรพสิ่ง (internet of things: IoT) ได้รับการออกแบบและนำเสนอเพื่อก้าวข้ามข้อจำกัดของ GPS โดยระบบระบุตำแหน่งภายในอาคารนี้ถูกออกแบบให้สามารถระบุตำแหน่งได้อย่างแม่นยำ และพัฒนาเป็นโปรแกรมประยุกต์ที่ใช้ระบุตำแหน่งบนโทรศัพท์เคลื่อนที่ โดยระบบที่นำเสนอแบ่งออกเป็น 3 ส่วนหลักๆ คือ 1. ส่วนนำเข้าข้อมูล ซึ่งจะรับค่า RSSI และส่งต่อไปยังส่วนประมวลผล 2. ส่วนประมวลผลข้อมูล จะรับข้อมูลค่า RSSI จากส่วนนำเข้าข้อมูล เพื่อนำมาวิเคราะห์และทำนายตำแหน่งภายในอาคาร และ 3. ส่วนส่งออกข้อมูล โดยจะส่งผลการทำนายตำแหน่งที่ได้จากส่วนประมวลผลแสดงกลับไปยังโปรแกรมประยุกต์บนโทรศัพท์เคลื่อนที่

1.2 วัตถุประสงค์

1) เพื่อศึกษาการระบุตำแหน่งภายในอาคารโดยใช้การวิเคราะห์ค่า RSSI จากเทคโนโลยี IoT

2) เพื่อศึกษาวิธีการวิเคราะห์ข้อมูลและพัฒนาเพื่อเพิ่มประสิทธิภาพของการระบุตำแหน่งภายในอาคาร

3) เพื่อสร้างโปรแกรมประยุกต์บนโทรศัพท์เคลื่อนที่ ที่สามารถระบุตำแหน่งภายในอาคารได้อย่างแม่นยำ

4) สามารถนำการระบุตำแหน่งภายในอาคารเพื่อนำไปต่อยอดสำหรับวัตถุประสงค์อื่น ๆ

1.3 ขอบเขตของปริญญานิพนธ์

ขอบเขตของปริญญานิพนธ์นี้ จะมุ่งเน้นเพื่อใช้ในการระบุตำแหน่งภายในอาคาร โดยการใช้ค่า RSSI จากเทคโนโลยี IoT มาทำการวิเคราะห์และระบุตำแหน่งด้วยอัลกอริทึมการเรียนรู้เชิงลึกและอื่น ๆ เพื่อพัฒนาเป็นโปรแกรมประยุกต์บนโทรศัพท์เคลื่อนที่ ที่สามารถใช้ระบุตำแหน่งภายในอาคารได้

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 โครงข่ายของสรรพสิ่ง (internet of things: IoT)

โครงข่ายของสรรพสิ่ง (internet of things: IoT) เป็นกรอบแนวคิดของระบบโครงข่ายที่รองรับการเชื่อมต่อกับอุปกรณ์หลากหลายชนิด ตั้งแต่คอมพิวเตอร์ โทรศัพท์เคลื่อนที่ อุปกรณ์โครงข่าย อุปกรณ์อิเล็กทรอนิกส์ เซนเซอร์ และวัตถุต่าง ๆ เข้าด้วยกัน อันเป็นผลให้ระบบต่าง ๆ สามารถติดต่อสื่อสารและทำงานร่วมกันได้อย่างเป็นอัตโนมัติ ทั้งยังเป็นผลให้มนุษย์สามารถเข้าถึงข้อมูลได้หลากหลายยิ่งขึ้น ควบคุมอุปกรณ์และระบบต่าง ๆ ได้อย่างมีประสิทธิภาพมากขึ้น เทคโนโลยี IoT เป็นผลสืบเนื่องมาจากการพัฒนาระบบอินเทอร์เน็ต ซึ่งมีวัตถุประสงค์เพื่อการสร้างโครงข่าย เพื่อเชื่อมโยงอุปกรณ์ที่มีมาตรฐานแตกต่างกันให้สามารถสื่อสารกันได้ โดย IoT จะเปิดโอกาสให้มีการเชื่อมต่อในรูปแบบที่หลากหลายมากยิ่งขึ้น และรองรับอุปกรณ์ที่พัฒนาโดยผู้ผลิตที่มีเทคโนโลยีแตกต่างกันมากกว่าเดิม ในปัจจุบันสามารถจัดกลุ่มการเชื่อมต่ออุปกรณ์ต่าง ๆ เข้ากับโครงข่ายอินเทอร์เน็ต ได้ตามรูปแบบดังต่อไปนี้

2.1.1 การเชื่อมต่อผ่านอุปกรณ์สื่อสารระยะสั้น (Short-Range Devices)

เป็นรูปแบบการเชื่อมต่ออุปกรณ์ในระยะสั้นมากโดยใช้กำลังส่งต่ำมาก เหมาะสำหรับการสื่อสารในพื้นที่ครอบคลุมขนาดเล็ก ซึ่งอยู่ในลักษณะการเชื่อมต่อระหว่างอุปกรณ์ (Peer-to-peer) หรือการเชื่อมต่อแบบโครงข่าย ตัวอย่างของการเชื่อมต่อในลักษณะดังกล่าว เช่น Wi-Fi, Bluetooth, Z-Wave และ ZigBee เป็นต้น

2.1.2 การเชื่อมต่อผ่านโครงข่ายโทรศัพท์เคลื่อนที่

เป็นรูปแบบการให้บริการที่มีพื้นที่ครอบคลุมกว้าง โดยอาศัยการเชื่อมต่ออุปกรณ์เครื่องลูกข่าย IoT เข้ากับโครงสร้างพื้นฐานของระบบโทรศัพท์เคลื่อนที่ที่มีอยู่แล้ว ตัวอย่างของการเชื่อมต่อในลักษณะดังกล่าว เช่น เทคโนโลยี NB-IoT และ LTE-M เป็นต้น

2.1.3 การเชื่อมต่อผ่านโครงข่ายระยะทางไกลโดยใช้พลังงานต่ำ (Low Power Wide Area Network: LPWAN)

เป็นรูปแบบการเชื่อมต่อผ่านโครงข่ายกำลังส่งต่ำบริเวณกว้าง โดยเน้นใช้งานในลักษณะการสื่อสารแบบแถบความถี่แคบ (narrow band) หรือ แถบความถี่แคบพิเศษ (ultra

narrow band) ที่มีอัตราการส่งข้อมูลต่ำมาก ประหยัดพลังงานมาก และมีราคาอุปกรณ์ต่อหน่วยที่ต่ำ ตัวอย่างของการเชื่อมต่อในลักษณะดังกล่าว เช่น LoRaWAN, SigFox และ Ingenu เป็นต้น

2.1.4 การเชื่อมต่อผ่านโครงข่ายสื่อสารดาวเทียม

เป็นรูปแบบการเชื่อมต่อที่เหมาะสมกับการใช้งานที่มีพื้นที่ ครอบคลุมการให้บริการที่กว้างมากแต่การเชื่อมต่อดังกล่าวจะมีระยะเวลาการตอบสนอง (Latency) ช้ากว่าการเชื่อมต่อรูปแบบอื่น ๆ เนื่องจากระยะเวลาที่สัญญาณเดินทางไป-กลับ ระหว่างอุปกรณ์สื่อสารภาคพื้นโลกและดาวเทียม [1]

2.2 คลื่นความถี่ย่าน 920 - 925 MHz

2.2.1 คลื่นความถี่ย่าน 920 - 925 MHz

คลื่นความถี่ย่าน 920 - 925 MHz เป็นคลื่นความถี่ในย่าน Ultra High Frequency (UHF) ความถี่ 300 - 3,000 MHz และเป็นส่วนหนึ่งของคลื่นความถี่ Industrial, Scientific and Medical band (ISM) ย่าน 915 MHz (ช่วงความถี่ตั้งแต่ 902 - 928 MHz) มีคุณสมบัติที่เหมาะสมต่อการสื่อสารด้วยเทคโนโลยีที่หลากหลายในปัจจุบัน โดยเฉพาะการสื่อสารในรูปแบบที่มีการเคลื่อนที่หรือการกระจายตัวของเครื่องลูกข่ายเป็นบริเวณกว้าง และไม่ได้มุ่งเน้นการใช้งานในลักษณะที่มีอัตราการส่งข้อมูลที่สูงมากนัก จึงมีการใช้งานอุปกรณ์สื่อสารในย่านนี้อย่างแพร่หลาย คลื่นความถี่ย่านดังกล่าวมีคุณสมบัติที่สำคัญต่อระบบสื่อสารไร้สาย ดังนี้

1) คุณสมบัติทางเทคนิคของคลื่นความถี่ส่งผลให้มีพื้นที่ครอบคลุมการใช้งานเป็นบริเวณกว้างเมื่อเทียบกับการใช้งานย่านความถี่ที่สูงกว่า ผู้ให้บริการการเชื่อมต่อโครงข่ายสื่อสารจึงไม่ต้องติดตั้งสถานีฐานมากเกินความจำเป็น ซึ่งเป็นการประหยัดต้นทุนการให้บริการของผู้ให้บริการ และยังช่วยลดผลกระทบที่จะเกิดเป็นภาระแก่ผู้ใช้บริการต่อไป

2) เป็นย่านความถี่ที่เหมาะสมต่อการออกแบบสายอากาศที่ติดตั้งภายในอุปกรณ์เชื่อมต่อให้มีขนาดเล็ก เนื่องจากหลักการออกแบบสายอากาศรับหรือส่งสัญญาณต้องอาศัยความยาวคลื่นเป็นปัจจัยสำคัญในการพิจารณา ซึ่งคลื่นความถี่สูงจะมีความยาวคลื่นน้อย ส่วนคลื่นความถี่ต่ำจะมีความยาวคลื่นมากกว่า ดังนั้นการใช้คลื่นความถี่ต่ำกว่าจึงจำเป็นต้องใช้สายอากาศที่มีความยาวหรือขนาดมากกว่าการใช้คลื่นความถี่สูง และอาจไม่เหมาะสมกับการใช้งานในลักษณะติดตั้งภายในอุปกรณ์ที่มีขนาดเล็ก

3) เป็นย่านความถี่ที่มีการรวมกลุ่มจัดทำมาตรฐานสำหรับอุปกรณ์และการเชื่อมต่อร่วมกันในระดับสากล ทำให้อุปกรณ์ที่มีมาตรฐานจากผู้ผลิตทั่วโลกสามารถใช้งานร่วมกันได้

4) เป็นย่านความถี่ที่ได้รับความนิยมอย่างแพร่หลาย โดยมีกลุ่มผู้ลงทุนและผู้ผลิตอุปกรณ์รายใหญ่รองรับและให้การสนับสนุนเป็นจำนวนมาก ทำให้เกิดทางเลือกในการเลือกใช้อุปกรณ์ที่เหมาะสมกับการให้บริการหรือการใช้งานในลักษณะต่าง ๆ และมีต้นทุนด้านอุปกรณ์ต่ำกว่าการใช้ย่านความถี่ที่มีกลุ่มผู้ผลิตให้การสนับสนุนน้อย

5) เป็นย่านความถี่ที่ถูกกำหนดให้สามารถใช้งานร่วมกันได้เป็นการทั่วไปในประเทศไทย ผู้ใช้งานจึงไม่จำเป็นต้องขอรับการจัดสรรคลื่นความถี่จากสำนักงานคณะกรรมการกิจการกระจายเสียง กิจการโทรทัศน์ และกิจการโทรคมนาคมแห่งชาติ (กสทช.) ซึ่งจะช่วยอำนวยความสะดวกและลดต้นทุนการใช้งานทั้งในด้านเวลา ขั้นตอน และค่าใช้จ่าย ยิ่งไปกว่านั้นผู้ใช้งานสามารถประยุกต์ใช้เทคโนโลยีดังกล่าวเพื่อวัตถุประสงค์ต่าง ๆ ได้ตามความเหมาะสมภายใต้เงื่อนไขหรือกฎระเบียบที่ กสทช. กำหนด โดยไม่มีการผูกมัดกับกิจการหรือวัตถุประสงค์ใดวัตถุประสงค์หนึ่งอีกด้วย

2.2.2 ประเภทของการใช้งานคลื่นความถี่ย่าน 920 - 925 MHz

การใช้คลื่นความถี่ย่าน 920 - 925 MHz ส่วนใหญ่เป็นการใช้งานในลักษณะแถบความถี่แคบ (narrow band) และอัตราการส่งข้อมูลต่ำ (low bit rate) โดยทั่วไปเป็นการใช้งานอุปกรณ์ระบุตัวตนด้วยคลื่นวิทยุหรือคลื่นความถี่ (Radio Frequency Identification: RFID) ร่วมกับอุปกรณ์สื่อสารระยะสั้น (Short-Range Radio-communication Devices: SRD) และอุปกรณ์สื่อสารกำลังส่งต่ำแบบวงกว้าง (Low-Power Wide-Area Network: LPWAN) ซึ่งอุปกรณ์ดังกล่าวล้วนเป็นที่นิยมสำหรับการนำมาประยุกต์ใช้งาน IoT ตามรูปแบบการใช้งานที่เหมาะสม เนื่องจากเทคโนโลยีการสื่อสารถูกพัฒนาไปอย่างรวดเร็ว ทำให้เกิดการประยุกต์ใช้งานอุปกรณ์ดังกล่าวในหลากหลายรูปแบบเพื่ออำนวยความสะดวกให้กับผู้ใช้งานจนกลายเป็นที่นิยมอย่างกว้างขวางในประเทศไทยปัจจุบัน ซึ่งเทคโนโลยีหลัก ๆ ที่เหมาะแก่การนำมาประยุกต์ใช้ในการระบุตำแหน่งภายในอาคารที่สำคัญ มีดังนี้

2.2.2.1 อุปกรณ์สื่อสารกำลังส่งต่ำแบบวงกว้าง (Low-Power Wide-Area Network: LPWAN)

LPWAN คือ การสื่อสารแบบไร้สายที่มีลักษณะเป็นโครงข่ายซึ่งมีระยะการสื่อสารเป็นวงกว้าง โดยใช้พลังงานต่ำทำให้อุปกรณ์ต่าง ๆ สามารถเชื่อมต่อและสื่อสารกันได้ระยะไกล เป็นคุณสมบัติที่ออกแบบมาเพื่อตอบสนองการใช้งาน IoT ที่มีการเชื่อมต่อของอุปกรณ์เป็นจำนวนมากหรือลักษณะการสื่อสารที่เรียกว่า mMTC (Massive Machine Type Communications) ที่อาศัยต้นทุนการผลิต การติดตั้ง และการซ่อมบำรุงต่ำ รวมถึงมีอายุการใช้งานที่ยาวนาน LPWAN มีคุณลักษณะเฉพาะที่เด่นชัด ดังนี้

1) รูปแบบการสื่อสารแบบไร้สายระยะไกล
2) อัตราการส่งข้อมูลต่ำ
3) ใช้พลังงานต่ำ
4) ต้นทุนการติดตั้งและซ่อมบำรุงต่ำ
5) อายุการใช้งานยาวนาน

2.2.2.2 IEEE 802.11ah (Wi-Fi HaLow)

เป็นมาตรฐานที่ถูกกำหนดขึ้นอย่างเป็นทางการในช่วงกลางปี ค.ศ. 2017 โดยสถาบันวิชาชีพ วิศวกรไฟฟ้าและอิเล็กทรอนิกส์ (The Institute of Electrical and Electronics Engineers: IEEE) เพื่อรองรับการใช้งาน IoT ต่าง ๆ ซึ่งใช้คลื่นความถี่ ISM ในย่าน UHF (920 - 925 MHz สำหรับประเทศไทย) มีจุดเด่นคือกินพลังงานน้อยลงกว่ามาตรฐาน Wi-Fi ในปัจจุบัน แต่สามารถส่งสัญญาณได้ไกลกว่าเกือบ 2 เท่า และเนื่องจาก HaLow มีความยาวคลื่นที่สูงกว่ามาตรฐาน Wi-Fi ปกติ ส่งผลให้มีอำนาจในการทะลุทะลวงกำแพงหรือสิ่งกีดขวางได้ดีกว่า จึงเหมาะสำหรับการใช้งานกับอุปกรณ์ประเภท IoT ซึ่งเป็นอุปกรณ์ขนาดเล็กและจำเป็นต้องเชื่อมต่อหากันตลอดเวลา โดย Wi-Fi HaLow มีคุณสมบัติที่เด่นแตกต่างจากเทคโนโลยีอื่น ๆ ที่ใช้คลื่นความถี่ย่านเดียวกันคือมีอัตรารับส่งข้อมูลที่สูงกว่า ซึ่งต้องอาศัยความกว้างแถบความถี่มากกว่าเทคโนโลยีอื่นด้วย แม้ว่าในปัจจุบันยังไม่ค่อยมีการใช้งานออกมาให้เห็น แต่ปริมาณการใช้งานอาจมีเพิ่มมากขึ้นตามความก้าวหน้าของเทคโนโลยีและการเจริญเติบโตของการประยุกต์และอุตสาหกรรมในอนาคต [2][3]

2.3 คลื่นความถี่ย่าน 2.4 GHz

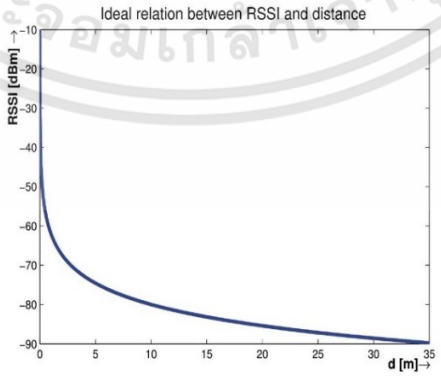
มาตรฐานของสัญญาณแลนไร้สาย (wireless LAN) สำหรับให้บริการอินเทอร์เน็ตในปัจจุบัน มีอยู่ 2 ย่านความถี่ คือ 2.4 GHz และ 5 GHz โดยย่านความถี่ 2.4 GHz จะมีจำนวนช่องสัญญาณทั้งหมด 11 ช่องสัญญาณสำหรับประเทศไทย ซึ่งย่านความถี่ 2.4 GHz จะมีลักษณะเด่นคือ สามารถส่งสัญญาณได้ในระยะไกล เนื่องจากเป็นคลื่นความถี่ต่ำ แต่มีข้อจำกัดทางด้านความเร็วในการรับ-ส่งข้อมูลระหว่างอุปกรณ์ และเนื่องจากการมีการเชื่อมต่อสัญญาณไร้สายเพิ่มมากขึ้นระหว่างอุปกรณ์ จึงส่งผลทำให้ช่องความถี่ย่าน 2.4 GHz เริ่มไม่เพียงพอต่อการใช้งาน จึงทำให้เกิดปัญหาช่องความทับซ้อน ซึ่งส่งผลให้การรับ-ส่งข้อมูลเกิดความล่าช้าขึ้น [4]

2.4 ค่าตัวชี้บอกความแรงของสัญญาณที่ได้รับ (Receive Signal Strength Indicator: RSSI)

RSSI เป็นค่าที่ใช้บอกความแรงของสัญญาณวิทยุที่ได้รับในเทอมของพลังงานมีหน่วยเป็น เดซิเบล - มิลลิวัตต์ (decibels milliwatt: dBm) โดยค่า RSSI จะแปรผันตรงกับความแรงของสัญญาณ ถ้าค่า RSSI มีค่ามากแสดงว่าสัญญาณที่ได้รับมีความแรงสูง นั่นคือตัวส่งและตัวรับอยู่ใกล้กัน และในทางกลับกันหากค่า RSSI มีค่าน้อยแสดงว่าสัญญาณที่ได้รับมีความแรงต่ำ คือตัวส่งและตัวรับอยู่ไกลกัน [5]

2.5 แบบจำลองการแพร่กระจายคลื่นวิทยุ

แนวคิดในการประมาณระยะทางจะสันนิษฐานจากกำลังสัญญาณที่ได้รับ (Received Signal Strength: RSS) ซึ่งเป็นฟังก์ชันของกำลังส่งและระยะทางระหว่างเครื่องส่งกับเครื่องรับดังรูปที่ 2.1



รูปที่ 2.1 ความสัมพันธ์ทางอุดมคติระหว่างค่า RSSI กับระยะทาง [6]

แบบจำลอง Path Loss ที่มักจะใช้กับระยะทางสั้น ๆ เช่น ในอาคาร ดังสมการที่ (2.1) [7]

$$\frac{P_r(d_0)}{P_r(d)} = \left(\frac{d}{d_0}\right)^{\eta} \tag{2.1}$$

เมื่อ $P_r(d)$ คือ RSS ที่ระยะ d ในหน่วยมิลลิวัตต์

$P_r(d_0)$ คือ RSS ที่ระยะอ้างอิง d_0 ในหน่วยมิลลิวัตต์

η คือ เลขชี้กำลัง Path Loss

d คือ ระยะทางระหว่างเครื่องส่งกับเครื่องรับในหน่วยเมตร

d_0 คือ ระยะทางอ้างอิงในหน่วยเมตร

แปลงหน่วยของกำลังให้อยู่ในรูปฟังก์ชันของ RSSI ดังสมการที่ (2.3)

$$RSSI = 10\log\frac{P}{0.001} \tag{2.2}$$

$$P = 10^{\left(\frac{RSSI-30}{10}\right)} \tag{2.3}$$

เมื่อ P คือกำลังสัญญาณในหน่วยมิลลิวัตต์

$RSSI$ คือกำลังสัญญาณในหน่วย dBm

แปลงสมการ Path Loss ให้อยู่ในรูปความสัมพันธ์ของ RSSI กับระยะทางดังสมการที่ (2.6) และ (2.7)

$$\log\left(P\left(d_0\right)\right) - \log\left(P\left(d\right)\right) = \eta\log\left(\frac{d}{d_0}\right) \tag{2.4}$$

$$\frac{RSSI_{d_0} - RSSI}{10} = \eta\log\left(\frac{d}{d_0}\right) \tag{2.5}$$

$$RSSI = RSSI_{d_0} - 10\eta\log\left(\frac{d}{d_0}\right) \tag{2.6}$$

$$d = d_0 10^{\left(\frac{RSSI_{d_0} - RSSI}{10\eta}\right)} \tag{2.7}$$

2.6 ปัญญาประดิษฐ์ (Artificial Intelligence: AI)

Artificial Intelligence (AI) หมายถึงเทคโนโลยีที่ทำการจำลองความฉลาดของมนุษย์ลงในเครื่องจักรที่ถูกตั้งโปรแกรมให้คิดและเลียนแบบเหมือนมนุษย์ โดยการพัฒนาระบบเพื่อให้มีความสามารถในการรับรู้ เรียนรู้ ใช้เหตุผล และตัดสินใจเลือกทางเลือกที่ดีสุดจากการวิเคราะห์ข้อมูลที่เกี่ยวข้องตามเงื่อนไขที่กำหนด โดยแบ่งออกเป็น 3 ประเภท

1) Artificial Narrow Intelligence (ANI) หรือ Weak AI หรือ Narrow AI คือ ปัญญาประดิษฐ์ที่มีระดับปัญญาและความสามารถในการเรียนรู้เฉพาะด้านใดเรื่องหนึ่ง

2) Artificial General Intelligence (AGI) หรือ Strong AI หรือ Deep AI คือ ปัญญาประดิษฐ์ที่มีระดับปัญญาและความสามารถเทียบเท่ากับมนุษย์ โดยสามารถเรียนรู้ได้หลายด้านเวลาเดียวกัน

3) Artificial Super Intelligence (ASI) คือปัญญาประดิษฐ์ที่มีระดับปัญญาและเหนือกว่ามนุษย์ โดยมีความสามารถที่หลากหลายรวมถึงการแก้ปัญหาและการตัดสินใจที่เหนือกว่ามนุษย์ [8]

2.6.1 การเรียนรู้ของเครื่อง (Machine Learning)

การเรียนรู้ของเครื่อง คือประเภทของปัญญาประดิษฐ์ ที่ระบบมีความสามารถในการเรียนรู้ได้ด้วยตัวเองจากข้อมูลที่มีอยู่เพื่อพัฒนากระบวนการแก้ปัญหาที่เหมาะสม ซึ่งไม่จำเป็นต้องมีการกำหนดคำสั่งไว้อย่างชัดเจน โดยวิธีการของการเรียนรู้ของเครื่อง ก็คือการนำชุดข้อมูลตัวอย่าง (Sample Data หรือ Dataset) ที่มีอยู่ไปสร้างเป็นแบบจำลอง (Model) เพื่อการทำนายผล โดยแบ่งออกเป็น 3 ประเภท

2.6.1.1 Supervised Learning

Supervised Learning เป็นกลุ่มแบบจำลองซึ่งใช้กับข้อมูลตัวอย่างที่มีคอลัมน์ผลลัพธ์สำหรับการทำนายผลหรือเรียกว่าคอลัมน์เป้าหมาย (Target) โดยแบ่งออกเป็น 2 ลักษณะคือ

1) Supervised Learning แบบการจำแนกประเภท (Classification) คือกรณีที่สามารถจำแนกค่าของคอลัมน์ผลลัพธ์ออกเป็นกลุ่มได้ เช่น True/False, Male/Female, Small/Medium/Large, 0/1, -1/1, 1/2/3/4

2) Supervised Learning แบบการถดถอย (Regression) คือกรณีที่คอลัมน์มีค่าเป็นตัวเลขที่ต่อเนื่องกันหลายค่า โดยไม่สามารถจำแนกเป็นกลุ่มได้

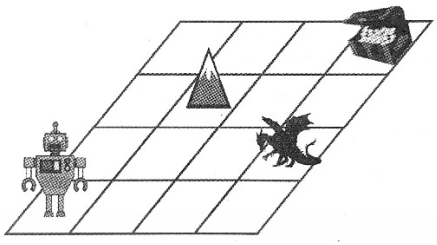
แบบจำลองซึ่งอยู่ในกลุ่ม Supervised Learning เช่น Linear Regression, Logistic Regression, Support Vector Machines, K-Nearest Neighbors, Naïve Bayes, Decision Tree, Neural Network รวมถึงแบบจำลองในกลุ่ม Ensemble Learning

2.6.1.2 Unsupervised Learning

Unsupervised Learning คือกลุ่มแบบจำลองที่ใช้สำหรับข้อมูลที่ไม่มีคอลัมน์ผลลัพธ์ หรือไม่มีเป้าหมายในการทำนายผลนั่นเอง ซึ่งไม่สามารถหาความสัมพันธ์ที่ชี้ชัดได้ว่าคอลัมน์ใดควรจะเป็นเป้าหมายของการทำนายผล โดยในกรณีนี้จะต้องใช้วิธีจัดกลุ่มข้อมูลหรือเรียกว่า การจับกลุ่ม (Clustering) แล้วหาจุดศูนย์กลางของแต่ละกลุ่มหรือจุดเซนทรอยด์ (Centroid) เพื่อใช้ตัวแทนของข้อมูลในกลุ่มนั้น ๆ ซึ่งแบบจำลองที่จัดอยู่ในกลุ่ม Unsupervised Learning เช่น K-Means Clustering เป็นต้น

2.6.1.3 Reinforcement Learning

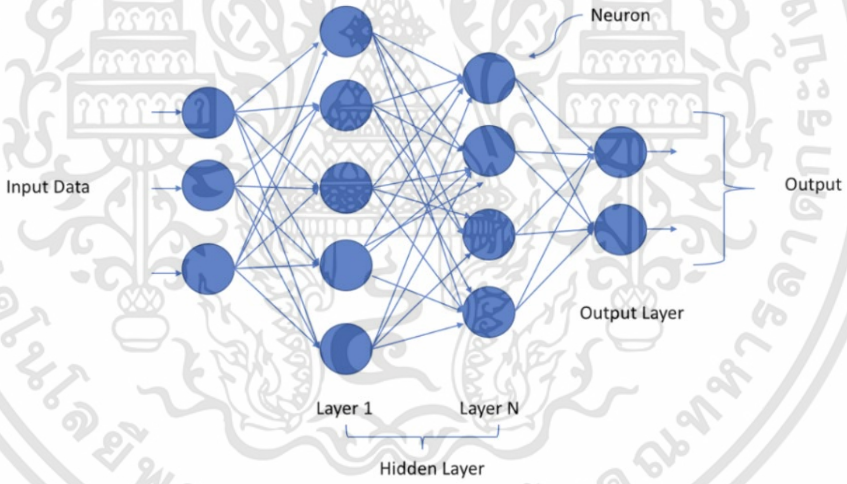
Reinforcement Learning จะแตกต่างไปจากการเรียนรู้ของเครื่องในกลุ่มอื่น ๆ เนื่องจากแบบจำลองประเภทนี้ไม่ใช่ข้อมูลตัวอย่างในการสร้างแบบจำลอง แต่จะตัดสินใจจากสภาพแวดล้อมที่เป็นอยู่ หรืออาจกล่าวได้ว่าข้อมูลส่งออกที่ได้ในขณะใด จะขึ้นอยู่กับข้อมูลนำเข้าในขณะนั้น ตัวอย่างที่ชัดเจนที่สุดคือ การนำการเสริมแรง (Reinforcement) ไปใช้ร่วมกับเกม เช่น สมมติว่าสร้างเกมที่มีลักษณะดังภาพที่ 2.2 จากภาพมีข้อกำหนดว่า ถ้าหุ่นยนต์เดินไปถึงช่องเก็บสมบัติตรงมุมขวาบนจะได้ 100 แต้ม ถ้าเดินไปเจอมังกรจะถูกตัด 500 แต้ม และหุ่นยนต์ไม่สามารถปีนภูเขาได้ นั่นแสดงว่า ก่อนที่หุ่นยนต์จะเคลื่อนที่เข้าไปในช่องใดจะต้องตรวจสอบก่อนว่าสามารถเดินข้ามไปได้หรือไม่และไม่ถูกตัดแต้ม ซึ่งลักษณะดังกล่าวนี้สามารถนำวิธีการของการเสริมแรง มาช่วยสำหรับการตัดสินใจก่อนจะเคลื่อนที่ในแต่ละครั้ง โดยการตรวจสอบจากสิ่งที่อยู่ล้อมรอบในขณะนั้น ๆ ทั้งนี้ หลักการของการเสริมแรง คือส่งข้อมูลนำเข้าออกไป แล้วตรวจสอบข้อมูลส่งออกดังกล่าว ซึ่งในปัจจุบันมีเทคโนโลยีเกี่ยวกับ AI หลายอย่างที่นำหลักการของการเสริมแรงไปประยุกต์ใช้งาน เช่น Game, Robot, Self-Driving เป็นต้น [9]



รูปที่ 2.2 เกมสมมติสำหรับอธิบายการเรียนรู้แบบเสริมแรง

2.6.2 การเรียนรู้เชิงลึก (Deep Learning)

การเรียนรู้เชิงลึก คือ ศาสตร์แขนงหนึ่งของการเรียนรู้ของเครื่อง ซึ่งมีวิธีการเรียนรู้แบบอัตโนมัติด้วยการเลียนแบบการทำงานของโครงข่ายประสาทของมนุษย์ (Neurons) โดยนำระบบโครงข่ายประสาทเทียม (Artificial Neural Network) ที่มีการซ้อนกันหลายชั้น (Layer) แสดงดังรูปที่ 2.3 ซึ่งการเรียนรู้เชิงลึกจะทำการเรียนรู้ข้อมูลตัวอย่าง โดยที่ข้อมูลดังกล่าวจะถูกนำไปใช้ในการตรวจจับรูปแบบหรือจัดหมวดหมู่ข้อมูล [10]



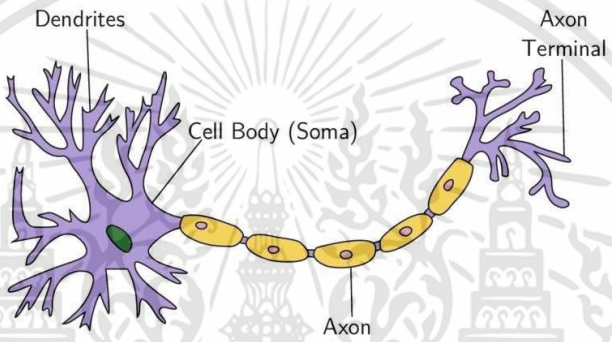
รูปที่ 2.3 โครงข่ายประสาทเทียมแบบหลายชั้น [11]

2.6.3 แบบจำลองโครงข่ายเซลล์ประสาท (Neural Network model)

โครงข่ายเซลล์ประสาท เป็นแบบจำลองหลักของการเรียนรู้ของเครื่องที่มักถูกเลือกใช้สำหรับงานด้าน AI จนอาจกล่าวได้ว่าโครงข่ายเซลล์ประสาทคือส่วนหนึ่งของ AI เพราะฉะนั้นผู้ที่จะเรียนรู้เพื่อก้าวไปสู่การสร้างปัญญาประดิษฐ์ก็จำเป็นต้องมีพื้นฐานเกี่ยวกับโครงข่ายเซลล์ประสาทที่ดีพอในระดับหนึ่ง

2.6.3.1 ลักษณะของแบบจำลองโครงข่ายเซลล์ประสาท

แบบจำลองโครงข่ายเซลล์ประสาทนี้ได้จำลองการทำงานมาจากระบบโครงข่ายประสาทจริงทางชีววิทยา (Biological Neural Network) ที่มีลักษณะดังรูปที่ 2.4 ซึ่งเป็นโครงสร้างของเซลล์ประสาท (Neuron) 1 อัน ซึ่งถ้านำหลาย ๆ เซลล์มารวมกันก็จะกลายเป็นโครงข่าย เพราะฉะนั้นอาจเรียกแบบจำลองนี้อีกอย่างว่า โครงข่ายเซลล์ประสาทเทียม (Artificial Neural Network: ANN) โดยจัดอยู่ในกลุ่ม Supervised Learning



รูปที่ 2.4 ระบบโครงข่ายประสาท [12]

2.6.3.2 องค์ประกอบของแบบจำลองโครงข่ายเซลล์ประสาท

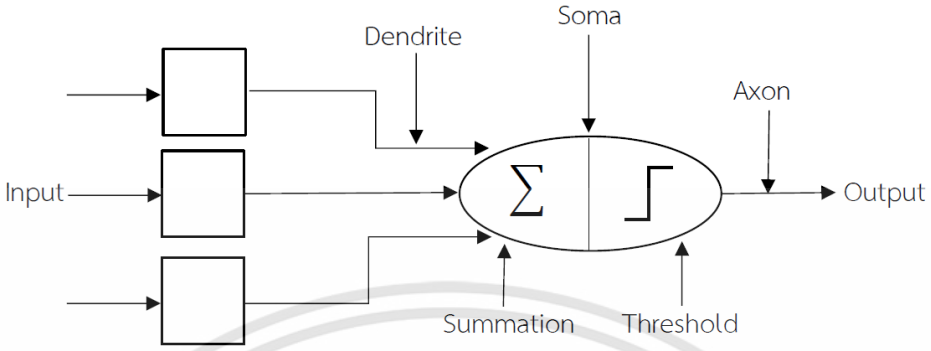
จากลักษณะของโครงข่ายเซลล์ประสาทดังรูปที่ 2.5 องค์ประกอบแต่ละอย่างมีหน้าที่พอสังเขปดังนี้

1) ใยประสาทนำเข้า (Dendrite) รับสัญญาณ (Signal) จากเซลล์ประสาทอื่น ๆ

2) เซลล์ร่างกาย (Cell Body หรือ Soma) รวมสัญญาณทั้งหมดที่รับมาเข้าด้วยกัน

3) ใยประสาทนำออก (Axon) เมื่อผลรวมของสัญญาณถึงระดับของค่าขีดเริ่มเปลี่ยน (Threshold) เซลล์ประสาทจะส่งสัญญาณผ่านใยประสาทนำออกไปยังเซลล์อื่น ๆ

4) จุดประสานประสาท (Synapse) เป็นจุดหรือปมที่ใช้เชื่อมต่อระหว่างเซลล์ประสาทอันหนึ่งกับเซลล์ประสาทอื่น ๆ



รูปที่ 2.5 องค์ประกอบของโครงข่ายเซลล์ประสาท

แบบจำลองโครงข่ายเซลล์ประสาทจะให้ผลการทำนายที่มีความถูกต้องแม่นยำสูง ดังนั้นงานด้าน AI ส่วนใหญ่จึงเลือกใช้แบบจำลองนี้เป็นหลักจนอาจกล่าวได้ว่า โครงข่ายเซลล์ประสาทก็คือส่วนหนึ่งของ AI เพราะฉะนั้นผู้ที่จะเรียนรู้เพื่อก้าวไปสู่การสร้างปัญญาประดิษฐ์ก็จำเป็นต้องมีพื้นฐานเกี่ยวกับโครงข่ายเซลล์ประสาทที่ดีพอในระดับหนึ่ง แต่อย่างไรก็ตาม โครงข่ายเซลล์ประสาทจัดเป็นการเรียนรู้เชิงลึกในระดับการเรียนรู้เชิงลึก จึงมีความซับซ้อนมากกว่าแบบจำลองอื่น ๆ ดังนั้นจึงอาจต้องใช้เวลาในการเรียนรู้นานพอสมควร

2.6.3.3 ข้อดีของแบบจำลองโครงข่ายเซลล์ประสาท

แบบจำลองโครงข่ายเซลล์ประสาทมีข้อดีที่น่าสนใจดังนี้

1) เป็นแบบจำลองหลักที่ใช้ในงานด้าน AI และงานค้นคว้าวิจัยอีกมากมาย

2) สามารถทำนายผลได้ทั้งแบบการถดถอยและการจำแนกประเภท

3) มีความถูกต้องแม่นยำและเชื่อถือได้

4) สามารถใช้ทำนายผลกับข้อมูลที่ไม่เป็นแบบเชิงเส้น (Non-Linear) ได้

2.6.3.4 ข้อเสียของแบบจำลองโครงข่ายเซลล์ประสาท

ข้อเสียของโครงข่ายเซลล์ประสาทที่ต้องพิจารณามีดังนี้

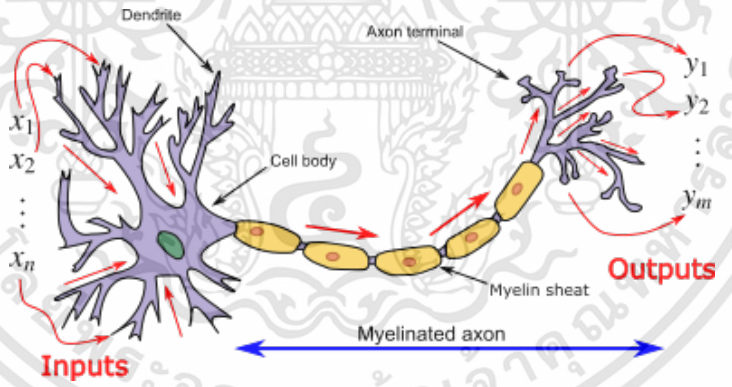
1) ในบางขั้นตอนมีวิธีการที่ค่อนข้างซับซ้อนดังนั้นจึงอาจใช้เวลาในการเรียนรู้พอสมควร

2) แบบจำลองจะมีความแม่นยำเมื่อมีข้อมูลสำหรับการฝึกฝนของแบบจำลองจำนวนมากพอ

3) มีลักษณะเป็นกล่องดำ (Black Box) นั่นคือไม่ทราบว่าตัวแปรอิสระมีผลต่อตัวแปรตามมากน้อยเพียงใด

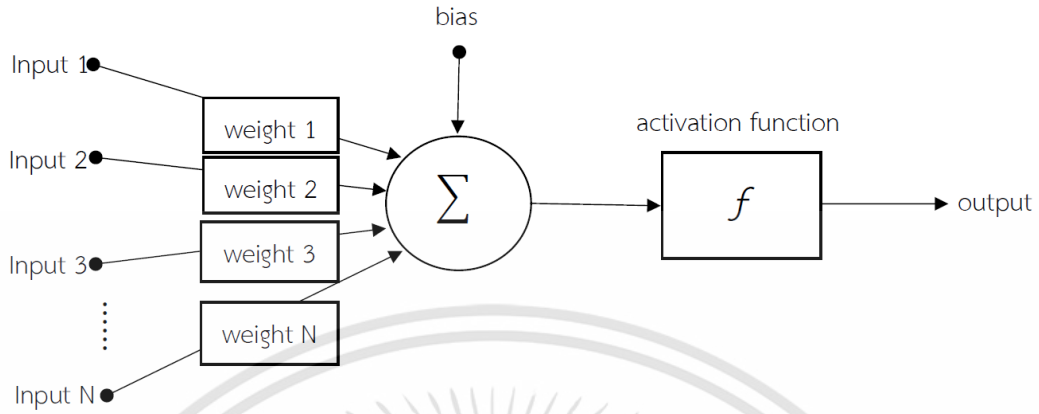
2.6.4 Single Layer Perceptron

Single Layer Perceptron หรือเรียกสั้น ๆ ว่า Perceptron คือหน่วยย่อยของโครงข่ายเซลล์ประสาท ซึ่งประกอบด้วยชั้นของเซลล์ประสาทเพียงอันเดียว ดังนั้นแหล่งข้อมูลบางแห่งอาจเรียกในชื่ออีกอย่างว่า Single Neuron และเมื่อนำเซลล์ประสาทหลาย ๆ อันมาเชื่อมต่อกันก็จะกลายเป็นโครงข่ายเซลล์ประสาท แต่ก่อนที่จะศึกษาในระดับโครงข่ายควรเริ่มต้นเรียนรู้จากส่วนย่อย ๆ ที่เรียกว่า Perceptron แสดงดังรูปที่ 2.6



รูปที่ 2.6 Single Layer Perceptron [13]

สำหรับการศึกษาในทางของการเรียนรู้ของเครื่องจะเขียน Perceptron ในรูปแบบไดอะแกรมที่มีส่วนประกอบพื้นฐานดังรูปที่ 2.7



รูปที่ 2.7 ไดอะแกรมส่วนประกอบพื้นฐานของ Perceptron

2.6.4.1 ข้อมูลนำเข้า (Input)

ข้อมูลนำเข้า คือข้อมูลที่นำไปประมวลผล ซึ่งก็มีลักษณะ (features) หรือตัวแปรอิสระ (x1, x2, x3, ...)

2.6.4.2 ค่าน้ำหนัก (Weight)

ค่าน้ำหนัก คือค่าน้ำหนักของข้อมูลนำเข้าแต่ละค่าซึ่งเป็นตัวบ่งชี้ว่าข้อมูลนำเข้าค่านั้น ๆ มีผลต่อข้อมูลส่งออกหรือผลลัพธ์มากน้อยเพียงใด ซึ่งถ้าเทียบกับสมการเส้นตรงก็คือค่าสัมประสิทธิ์ (Coefficient)

2.6.4.3 ไบแอส (Bias)

ไบแอส คือค่าที่กำหนดเพิ่มเติมลงไปซึ่งถ้าเทียบกับสมการเส้นตรงก็คือจุดตัดแกน Y

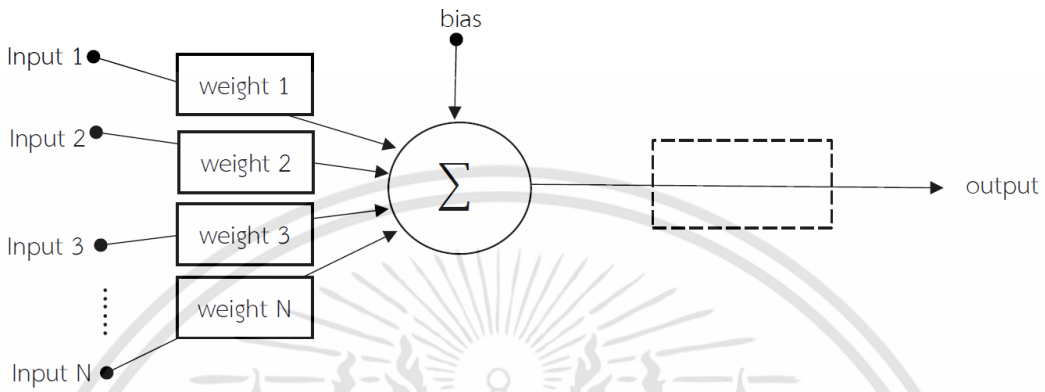
2.6.4.4 ฟังก์ชันการกระตุ้น (Activation Function)

ฟังก์ชันการกระตุ้น เป็นฟังก์ชันสำหรับการแปลงให้ได้ผลลัพธ์ตามรูปแบบที่ต้องการ

2.6.4.5 ข้อมูลส่งออก (Output)

ข้อมูลส่งออก คือผลลัพธ์สุดท้ายของการคำนวณหรือถ้าพิจารณาในแบบของการเรียนรู้ของเครื่องก็คือผลการทำนาย

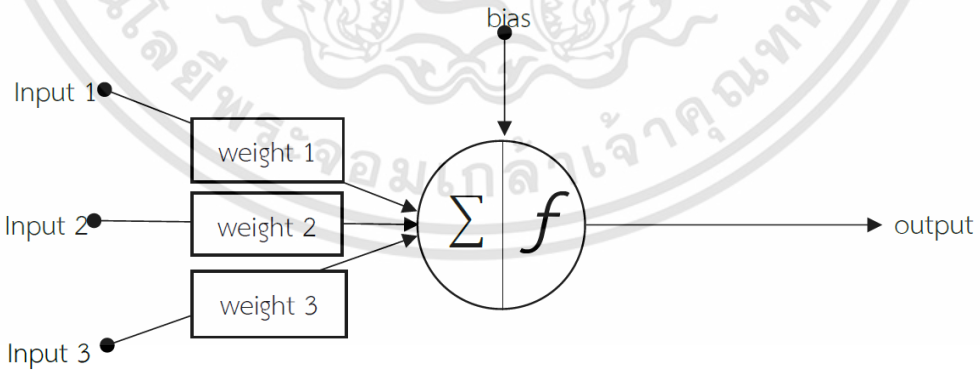
โดยไดอะแกรมส่วนประกอบพื้นฐานของ Perceptron แบบไม่มีฟังก์ชันการกระตุ้น แสดงดังรูปที่ 2.8



รูปที่ 2.8 ไดอะแกรมส่วนประกอบพื้นฐานของ Perceptron แบบไม่มีฟังก์ชันการกระตุ้น

2.6.5 ฟังก์ชันการกระตุ้น (Activation Function)

ฟังก์ชันการกระตุ้น หรือ ฟังก์ชันส่งผ่าน (transfer function) ใช้ในการแปลงข้อมูลส่งออกที่ได้จากเซลล์ประสาท (หลังการคูณข้อมูลนำเข้ากับน้ำหนักแล้วบวกด้วยค่าไบแอส) ให้ตรงตามลักษณะผลลัพธ์ที่ต้องการ ซึ่งในกรณีของ Single Layer Perceptron มีลำดับการทำงานดังรูปที่ 2.7 หรือในกรณีที่เขียนไดอะแกรมแบบย่อก็อาจรวมส่วนของฟังก์ชันการกระตุ้นไว้ในเซลล์ประสาท ดังรูปที่ 2.9

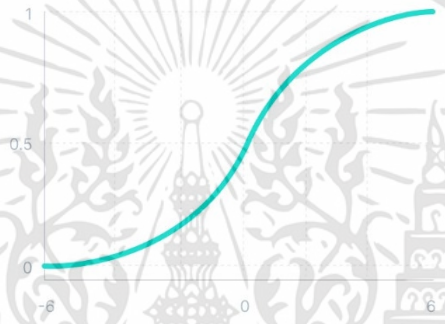


รูปที่ 2.9 ไดอะแกรมแบบย่อของลำดับการทำงานของ Perceptron

ฟังก์ชันการกระตุ้นมีให้เลือกใช้หลายแบบตามลักษณะผลลัพธ์ที่ต้องการ แต่มีเพียงบางฟังก์ชันเท่านั้นที่นิยมใช้งานกันเป็นส่วนใหญ่ในปัจจุบัน โดยมีรายละเอียดดังนี้

2.6.5.1 Sigmoid Function (หรือ Logistic Function) จะให้ผลลัพธ์ที่มีค่าอยู่ระหว่าง 0 – 1 ตามสมการที่ (2.8) และแสดงกราฟดังรูปที่ 2.10

$$y = \frac{1}{1+e^{-x}} \tag{2.8}$$



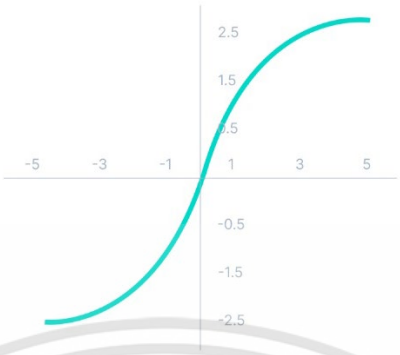
รูปที่ 2.10 Sigmoid Function [14]

สำหรับค่า e สามารถใช้ฟังก์ชัน math.exp() ของ Python หรือฟังก์ชัน np.exp() ของ NumPy เช่น

```
y = 1 / (1 + math.exp(-x))
z = 1 / (1 + np.exp(-x))
```

2.6.5.2 Hyperbolic Tangent ฟังก์ชันนี้จะให้ผลลัพธ์ที่มีค่าระหว่าง -1 ถึง 1 ตามสมการที่ (2.9) และแสดงกราฟดังรูปที่ 2.11

$$y = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.9}$$



รูปที่ 2.11 Hyperbolic Tangent Function [14]

สำหรับค่า e สามารถใช้คำสั่ง math.exp() ของ Python หรือใช้ฟังก์ชัน np.tanh() ของ NumPy โดยตรงได้ เช่น

```
y = (math.exp(x) - math.exp(-x))/(math.exp(x) + math.exp(-x))
z = np.tanh(x)
```

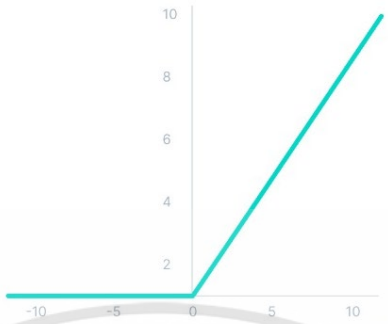
2.6.5.3 Rectifier Linear Unit (ReLu) ลักษณะผลลัพธ์ของฟังก์ชันนี้คือ

$y = x$ ถ้า $x > 0$ (2.10)

$y = 0$ ถ้า $x \leq 0$ (2.11)

ลักษณะของกราฟแสดงดังรูปที่ 2.12 ซึ่งสามารถหาค่าโดยใช้ฟังก์ชัน max() ของ Python หรือ np.maximum() ของ NumPy เช่น

```
y = max(0, x)
z = np.maximum(0, x)
```



รูปที่ 2.12 ReLU Function [14]

2.6.5.4 Leaky ReLU ลักษณะผลลัพธ์ของฟังก์ชันนี้คือ

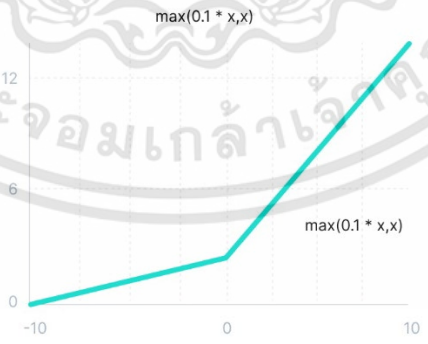
$y = x$ ถ้า $x > 0$ (2.12)

$y = ax$ ถ้า $x \leq 0$ (2.13)

เมื่อ a เป็นตัวเลขที่มีค่าน้อย ๆ เช่น 0.1

ลักษณะของกราฟแสดงดังรูปที่ 2.13 ซึ่งสามารถหาค่าโดยใช้ฟังก์ชัน max() ของ Python หรือ np.maximum() ของ NumPy เช่น

```
y = max(ax, x)
z = np.maximum(ax, x)
```

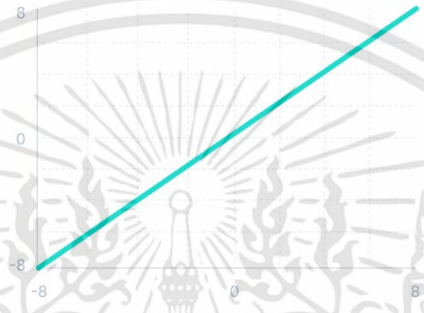


รูปที่ 2.13 Leaky ReLU Function [14]

2.6.5.5 Linear Function ลักษณะกราฟของฟังก์ชันนี้แสดงดังรูปที่ 2.14 ซึ่งมีฟังก์ชันดังสมการที่ (2.14) และ (2.15)

$$y = x \tag{2.14}$$

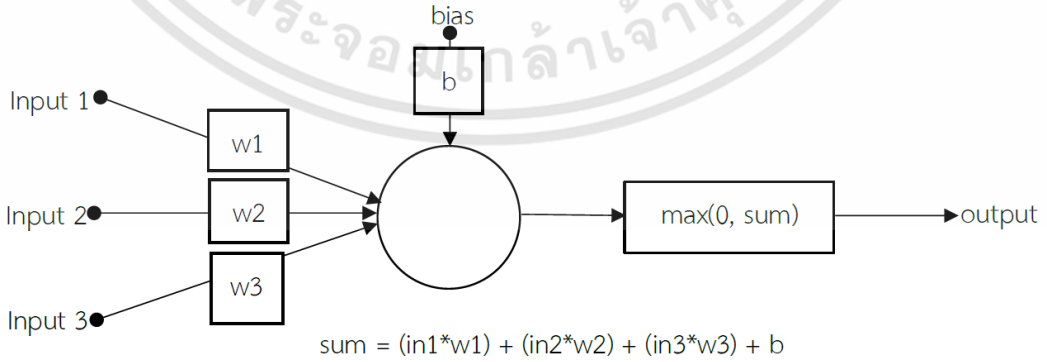
$y = ax$ ถ้า a คือค่าสัมประสิทธิ์ (2.15)



รูปที่ 2.14 Linear Function [14]

สำหรับหลักการเลือกฟังก์ชันการกระตุ้นนั้นก็ให้พิจารณาจากลักษณะผลลัพธ์ที่ต้องการเป็นหลักแต่อย่างไรก็ตามสำหรับกรณีของ Single Layer Perceptron ในปัจจุบันนิยมใช้ ReLU เป็นส่วนใหญ่

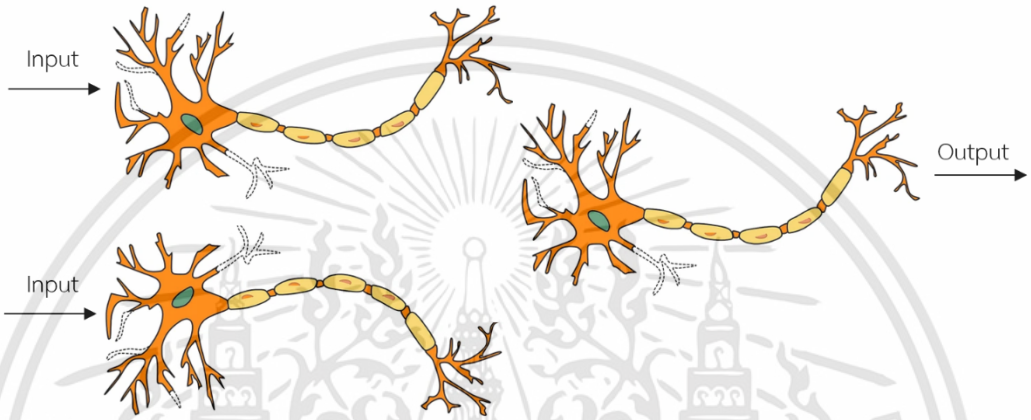
ส่วนแนวทางการใช้ฟังก์ชันการกระตุ้นทั้งหมดดังที่กล่าวมาก็คือหลังจากที่คำนวณผลรวมของผลคูณข้อมูลนำเข้าและน้ำหนัก แล้วบวกกับค่าไบแอส ก็นำมาดำเนินการต่อโดยกำหนดค่าให้แก่ฟังก์ชันการกระตุ้นแบบใดแบบหนึ่งตามที่เลือก แล้วค่าที่ได้จากฟังก์ชันดังกล่าวก็คือผลการทำนายของ Perceptron นั้น ดังแนวทางในรูปที่ 2.15



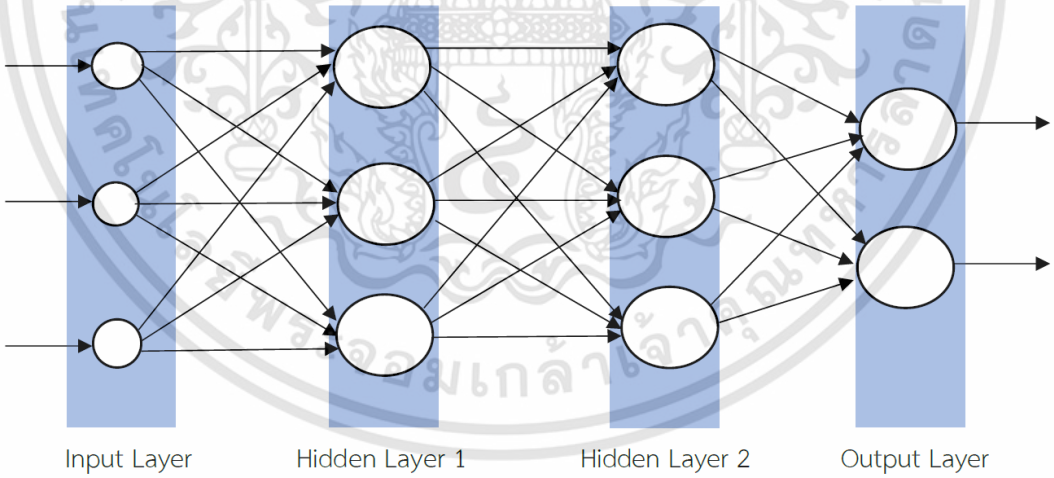
รูปที่ 2.15 แนวทางของผลทำนายของ Perceptron ที่ได้จากฟังก์ชันการกระตุ้น

2.6.6 Multi-layer Perceptron (MLP)

Multi-layer Perceptron (MLP) ก็คือการนำ Single Layer Perceptron หลาย ๆ อัน มาเชื่อมต่อกันเป็นโครงข่าย หรืออาจกล่าวได้ว่า MLP ก็คือรูปแบบที่แท้จริงของโครงข่ายเซลล์ประสาทนั้น ดังลักษณะในรูปที่ 2.16 และ 2.17



รูปที่ 2.16 Multi-layer Perceptron (1)



รูปที่ 2.17 Multi-layer Perceptron (2)

โครงข่ายเซลล์ประสาทเทียมจะแบ่งโครงสร้างออกเป็น 3 ชั้น คือ

1) Input Layer เป็นชั้นที่ใช้ในการรับค่าข้อมูลนำเข้าจากภายนอกหรือรับมาจากชั้น Output Layer อื่น ๆ โดยแต่ละข้อมูลนำเข้าก็จะมีค่าน้ำหนักของตัวเองเพื่อบ่งชี้ระดับความสำคัญที่มีผลต่อข้อมูลส่งออกเช่นเดียวกับ Single Layer Perceptron

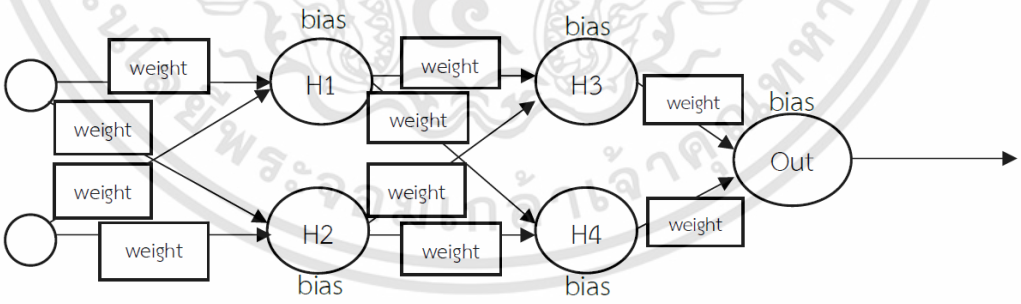
2) Hidden Layer เป็นชั้นที่อยู่ระหว่าง Input Layer และ Output Layer โดยรับค่าจากข้อมูลนำเข้ามาประมวลผลแล้วส่งออกไปยังชั้น Output Layer โดยภายในโครงข่ายเซลล์ประสาทอาจมีชั้น Hidden Layer ได้มากกว่า 1 ชั้น และในแต่ละชั้นก็อาจมีจำนวนเซลล์ประสาทได้มากกว่า 1 เซลล์ประสาท

3) Output Layer เป็นชั้นที่ใช้รับค่าของผลลัพธ์ที่ส่งมาจากชั้น Hidden Layer โดยในชั้นนี้อาจมีมากกว่า 1 เซลล์ประสาท

เนื่องจาก Multi-layer Perceptron (MLP) ก็เหมือนกับการนำ Single Layer Perceptron มาเชื่อมต่อกันดังนั้นวิธีการคำนวณก็ยังคงใช้แนวทางเดิม แต่อย่างไรก็ตามในกรณีของ MLP มีข้อมูลปลีกย่อยของส่วนต่าง ๆ ที่ต้องกำหนดเพิ่มเติม ดังรูปที่ 2.18 คือ

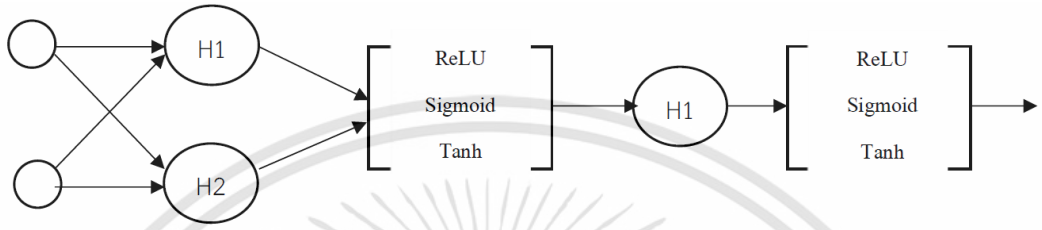
1) ข้อมูลนำเข้าแต่ละค่าที่ส่งมายังชั้น Hidden Layer และชั้น Output Layer ก็จะมีค่าน้ำหนักของตัวเอง

2) แต่ละเซลล์ประสาทในชั้น Hidden Layer และชั้น Output Layer จะมีค่าไบแอสแยกกันหรือกล่าวได้ว่าต้องมีค่าไบแอสสำหรับเซลล์ประสาทแต่ละอัน



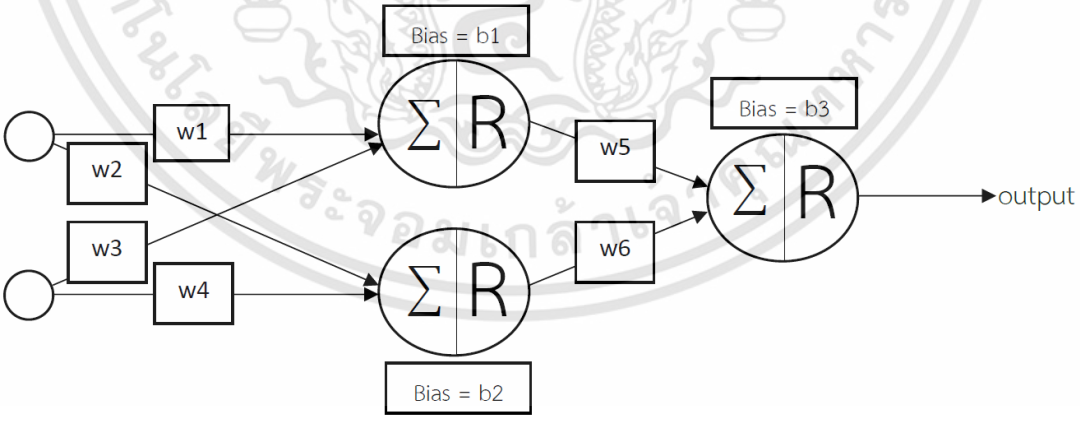
รูปที่ 2.18 ข้อมูลปลีกย่อยที่ต้องกำหนดในกรณีของ MLP

3) หลังจากการคำนวณ summation ของแต่ละเซลล์ประสาทในชั้น Hidden Layer และชั้น Output Layer จะต้องแปลงค่าที่ได้ด้วยฟังก์ชันการกระตุ้นอันใดอันหนึ่ง ดังที่ทำใน Single Layer Perceptron แสดงดังรูปที่ 2.19



รูปที่ 2.19 การแปลงค่าที่ได้ด้วยฟังก์ชันการกระตุ้น

จากทั้งหมดที่กล่าวมาจะเห็นได้ว่า MLP ประกอบด้วยข้อมูลปลีกย่อยในแต่ละเซลล์ประสาทและการเชื่อมโยงรวมถึงฟังก์ชันการกระตุ้น ซึ่งต้องนำมาคำนวณในทุกขั้นตอนกว่าจะได้ผลลัพธ์หรือผลการทำนาย ดังนั้นจึงค่อนข้างซับซ้อนกว่า Single Layer Perceptron แต่ทั้งนี้ก็ขึ้นอยู่กับจำนวนชั้นและจำนวนเซลล์ประสาทที่มีในแต่ละชั้น เพื่อให้เข้าใจง่ายขึ้นจะอธิบายลำดับขั้นตอนการคำนวณแต่ละค่าโดยตรงก่อน สมมติว่ามี MLP ที่ประกอบด้วยข้อมูลนำเข้า ค่าน้ำหนัก ค่าไบแอส ดังรูปที่ 2.20 และเลือกใช้ ReLU(R) เป็นฟังก์ชันการกระตุ้นทั้งหมด มีขั้นตอนการคำนวณดังนี้



รูปที่ 2.20 MLP ที่ประกอบด้วยข้อมูลนำเข้า ค่าน้ำหนัก และค่าไบแอส

1) หาค่า summation ของแต่ละเซลล์ประสาทในชั้น Hidden Layer โดยเอาค่าข้อมูลนำเข้าคูณค่าน้ำหนักบวกไบแอส แล้วแปลงด้วยฟังก์ชันการกระตุ้น

2) ผลลัพธ์ที่ได้จากชั้น Hidden Layer จะกลายเป็นข้อมูลนำเข้าของชั้น Output Layer ถัดไปคือการหาค่า summation ของแต่ละเซลล์ประสาทในชั้น Output Layer โดยเอาค่าข้อมูลนำเข้าคูณน้ำหนัก บวกไบแอส แล้วแปลงด้วยฟังก์ชันการกระตุ้นเหมือนเดิม แล้วค่าที่ได้ก็เป็นผลลัพธ์สุดท้าย

อย่างไรก็ตาม ไม่จำเป็นต้องเลือกใช้ฟังก์ชันการกระตุ้นแบบเดียวกันทั้งหมดก็ได้ แต่สามารถใช้หลายอย่างร่วมกันเช่น ในชั้น Hidden Layer อาจเลือกใช้แบบ ReLU แต่ในชั้น Output Layer ถ้าเป็นการทำนายผลแบบการจำแนกประเภทอาจเลือกใช้ Sigmoid หรือไม่ก็ Hyperbolic Tangent แต่หากจะทำนายผลแบบการถดถอยอาจเลือก ReU / Leaky ReLU หรือ Linear Function เพื่อให้ได้ผลลัพธ์ดังกล่าว เป็นต้น

2.6.7 หลักการของ Backpropagation

แบบจำลองจะมีความถูกต้องแม่นยำมากที่สุด เมื่อมีความผิดพลาดน้อยที่สุด ดังนั้นจึงเกิดปัญหาตามมาว่าทำอย่างไรจึงจะลดข้อผิดพลาดให้เหลือน้อยที่สุด ซึ่งค่าความผิดพลาดที่กล่าวมานี้จะดูจากข้อแตกต่างระหว่างผลลัพธ์จริง (ในข้อมูลตัวอย่าง) กับค่าที่ทำนายผลจากแบบจำลอง ดังลักษณะในตารางที่ 2.1

ตารางที่ 2.1 ผลลัพธ์จริงกับค่าที่ทำนายผลจากแบบจำลอง

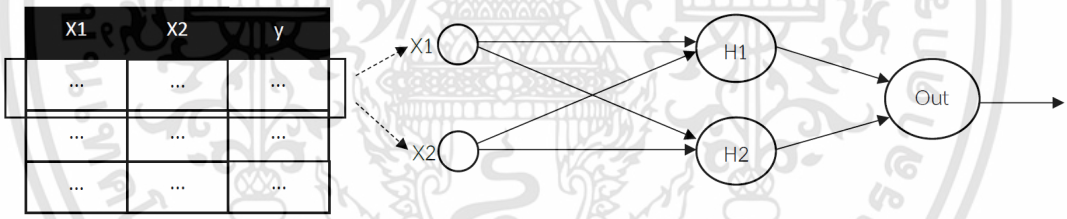
x1	x2	x3	y_actual	y_predict	error
...	...	...	20	5	15
...	...	...	10	10	0
...	...	...	15	15	0

การคำนวณในหัวข้อที่ผ่าน ๆ มานั้นทั้งกรณีของ Single Layer Perceptron และ Multi-layer Perceptron มีรูปแบบของสมการที่ใช้แสดงดังสมการที่ (2.16) เมื่อ y คือข้อมูลส่งออก x_n คือข้อมูลนำเข้า และ w_n คือค่าน้ำหนัก

$$y = x_1 w_1 + x_2 w_2 + x_3 w_3 + ... + x_n w_n + bias \tag{2.16}$$

ถึงแม้จะหาข้อมูลส่งออกในขั้นตอนสุดท้ายออกมาได้ก็ตาม แต่นั่นอาจไม่ใช่ผลลัพธ์ที่ถูกต้องที่สุด ทั้งนี้ก็เพราะว่าข้อมูลต่าง ๆ ที่นำมาใช้นั้นเป็นเพียงการสมมติขึ้นมาเท่านั้น ซึ่งหากพิจารณาในรูปการใช้งานจริงค่าข้อมูลนำเข้าเริ่มแรก จะใช้ข้อมูลตัวอย่างที่นำมาฝึกฝนแบบจำลอง แต่ค่าน้ำหนักจำเป็นต้องสมมติขึ้นมาเอง ดังนั้นต้องพิจารณาค่าน้ำหนักที่สมมติขึ้นมานั้นใกล้เคียงกับค่าที่แท้จริงเพียงใด โดยถ้าหากใกล้เคียงกับน้ำหนักจริงข้อผิดพลาดก็จะมีค่าน้อย แต่ถ้าค่าน้ำหนักที่สมมตินั้นแตกต่างจากค่าน้ำหนักจริงมากข้อผิดพลาดก็จะมีค่ามากตามไปด้วย ซึ่งก็แสดงว่าหากต้องการให้แบบจำลองมีความแม่นยำในการทำนายผลมากที่สุดก็ต้องคำนวณหาค่าน้ำหนักที่มีความถูกต้องหรือใกล้เคียงกับค่าที่แท้จริงให้มากที่สุด เพื่อลดข้อผิดพลาดให้เหลือน้อยที่สุด และนี่ก็คือขั้นตอนซึ่งมีความยุ่งยากต่อการเรียนรู้มากที่สุดของโครงข่ายเซลล์ประสาท สำหรับการหาค่าน้ำหนักของโครงข่ายเซลล์ประสาท มีแนวทางโดยสังเขปดังนี้

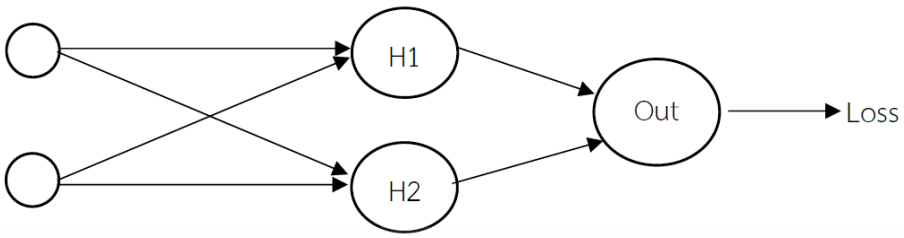
- เลือกข้อมูลตัวอย่างมา 1 แถวแล้วกำหนดเป็นค่าที่ชั้น Input Layer ส่วนค่าน้ำหนักและไบแอสก็ให้สมมติโดยการสุ่มตัวเลขขึ้นมา จากนั้นทำการคำนวณไปตามขั้นตอนดังที่ได้ศึกษามาจนกว่าจะได้ผลลัพธ์สุดท้ายหรือผลการทำนายออกมา ดังรูปที่ 2.21



รูปที่ 2.21 ขั้นตอนในการหาผลการทำนาย

- หาความแตกต่างระหว่างผลลัพธ์จริง (Actual) กับผลการทำนาย (Predict) หรือเรียกว่า Loss แสดงดังรูปที่ 2.22 เนื่องจากชั้น Output Layer อาจมีมากกว่า 1 เซลล์ประสาท ดังนั้นจึงใช้วิธีการค่าเฉลี่ยในแบบ Mean Square Error (MSE) ตามสมการที่ (2.17) เมื่อ n คือจำนวนข้อมูลส่งออกหรือจำนวนเซลล์ประสาทของชั้น Output Layer

$$Loss = \frac{1}{n}\sum (y_{actual} - y)^2 \qquad (2.17)$$



รูปที่ 2.22 การหาค่า Loss

- เนื่องจากไม่ทราบว่าค่าน้ำหนักที่แท้จริงเป็นเท่าใดดังนั้นจึงใช้หลักการของ Gradient Descent เพื่อหาค่าน้ำหนักที่ทำให้ข้อผิดพลาดมีค่าน้อยที่สุด โดยเทียบระหว่างค่า Loss ที่เกิดขึ้นกับน้ำหนักแต่ละค่าในรูปแบบของ Partial Derivative ดังเช่นสมการที่ (2.18)

$$\frac{\partial L}{\partial w_n} \tag{2.18}$$

เมื่อ w_n คือค่าน้ำหนัก และ L คือค่าความผิดพลาด หมายความว่าหากค่า w_5 เปลี่ยนแปลงไปโดยที่น้ำหนักค่าอื่น ๆ เช่น w_1, w_2, ... คงที่ จะมีผลให้ค่าความผิดพลาดหรือ Loss เพิ่มขึ้นหรือลดลง ซึ่งจะต้องทำเช่นนี้กับค่าน้ำหนักอื่น ๆ ที่เหลือจนครบทั้งหมด

- เนื่องจากมีการคำนวณค่า Loss ที่ชั้น Output Layer ดังนั้นการหาค่า Gradient ต้องเริ่มต้นที่ค่าของข้อมูลส่งออกแล้วย้อนกลับไปยังชั้นก่อนนี้ตามลำดับจนถึงชั้น Input Layer โดยใช้วิธี Partial Derivative ในรูปแบบของกฎลูกโซ่ (Chain Rule) ซึ่งมีลักษณะทั่วไปตามสมการที่ (2.19) และ (2.20)

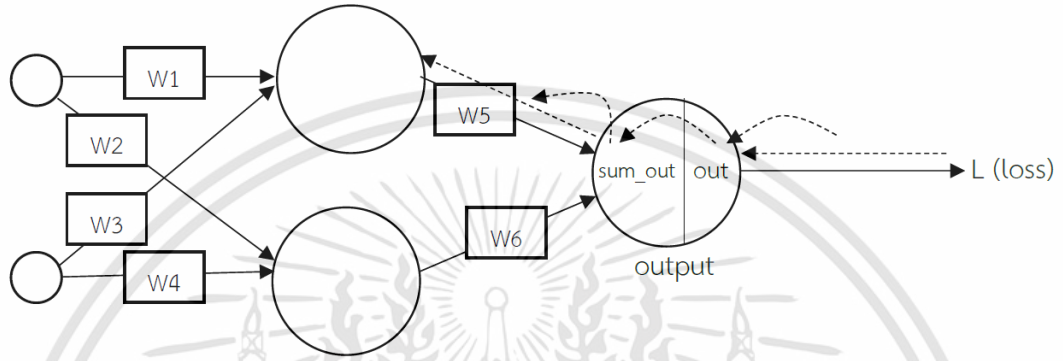
$$\frac{\partial a}{\partial c} = \frac{\partial a}{\partial b} \cdot \frac{\partial b}{\partial c} \tag{2.19}$$

$$\frac{\partial a}{\partial z} = \frac{\partial a}{\partial x} \cdot \frac{\partial x}{\partial y} \cdot \frac{\partial y}{\partial z} \tag{2.20}$$

- เนื่องจากการหาค่า Gradient จะดำเนินการแบบย้อนกลับจากชั้น Output Layer ไปยังชั้น Input Layer ดังนั้นจึงเรียกขั้นตอนนี้ว่า Backpropagation ซึ่งในรูปที่ 2.23 2.24 และ 2.25 เป็นเพียงหลักการในบางขั้นตอนโดย sum_ เป็นผลรวมระหว่างข้อมูลนำเข้า

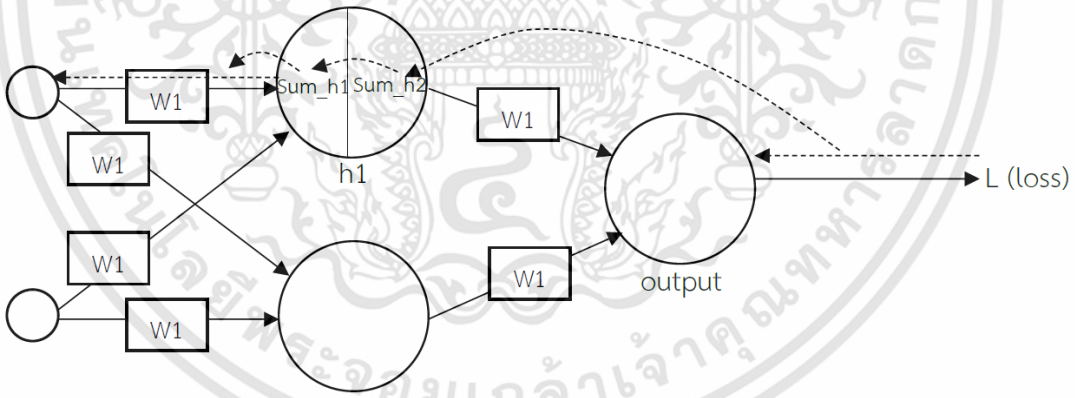
คูณด้วยค่าน้ำหนักบวกไบแอส ในแบบ Forward ส่วน out_ เป็นผลลัพธ์หลังการคำนวณฟังก์ชันการกระตุ้นแล้ว

$$\frac{\partial L}{\partial w5} = \frac{\partial L}{\partial out} \cdot \frac{\partial out}{\partial sum\_out} \cdot \frac{\partial sum\_out}{\partial w5}$$

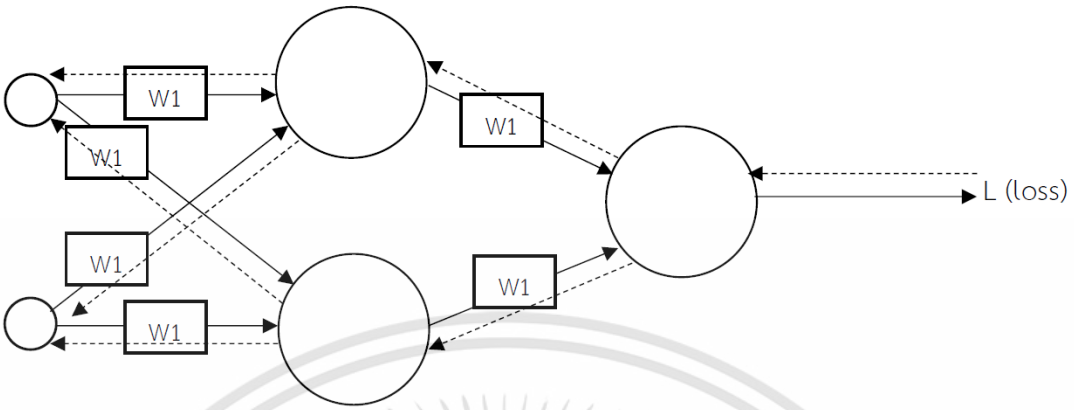


รูปที่ 2.23 ขั้นตอนในการทำ Backpropagation (1)

$$\frac{\partial L}{\partial w1} = \frac{\partial L}{\partial out\_h1} \cdot \frac{\partial out\_h1}{\partial sum\_h1} \cdot \frac{\partial sum_{h1}}{\partial w1}$$



รูปที่ 2.24 ขั้นตอนในการทำ Backpropagation (2)



รูปที่ 2.25 ขั้นตอนในการทำ Backpropagation (3)

- ต้องนำข้อมูลรายการอื่น ๆ มาทำเช่นเดียวกัน แล้วอัปเดตค่าน้ำหนักไปเรื่อย ๆ จนกว่าจะได้ค่าน้ำหนักซึ่งให้ผลลัพธ์ที่มีความถูกต้องมากที่สุด หรือมีข้อผิดพลาดน้อยที่สุดสำหรับข้อมูลชุดนั้น

สิ่งที่ได้กล่าวมานี้เป็นเพียงแนวทางโดยสังเขปเท่านั้น ความจริงยังมีลักษณะปลีกย่อยอื่น ๆ ที่ต้องดำเนินการอีกมาก ซึ่งมองเห็นถึงความยากลำบากที่จะต้องเจอ ด้วยเหตุนี้ขั้นตอนของ Backpropagation จึงเป็นงานที่ยุ่งยากเกินไปดังนั้นโดยส่วนใหญ่นิยมใช้ไลบรารีมากกว่าที่จะคำนวณเอง [9]

2.6.8 Optimizer

Optimizer เป็นอัลกอริทึมที่ใช้สำหรับการปรับค่าน้ำหนัก และค่าไบแอสที่ใช้ในการฝึกฝนโครงข่ายประสาทเทียมของการเรียนรู้เชิงลึก [15]

2.6.8.1 Gradient Descent

สำหรับหลักการของ Gradient Descent เริ่มแรกคือทำการสุ่มค่าน้ำหนัก จากนั้นทำการปรับค่า Gradient ให้ค่าน้ำหนักเปลี่ยนดังสมการที่ (2.21) และมีลักษณะการเปลี่ยนแปลงของฟังก์ชันดังรูปที่ 2.26 [16]

$$w_n = w_0 - \eta \frac{\partial E}{\partial w_0} \tag{2.21}$$

เมื่อ w_0 คือ ค่าน้ำหนักปัจจุบัน

η คือ อัตราการเรียนรู้

$\frac{\partial E}{\partial w_0}$ คือ ค่า Gradient ปัจจุบัน

สำหรับค่าผิดพลาดที่ใช้เป็นค่า Mean Square Error ดังสมการที่ (2.22)

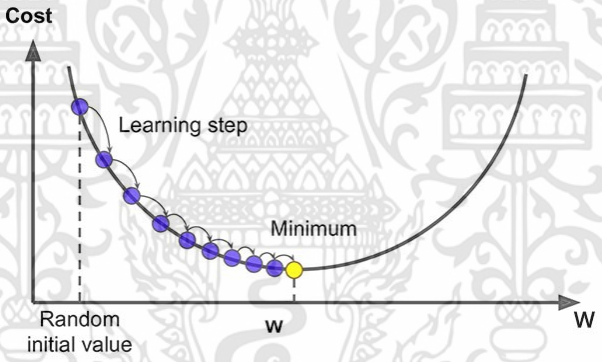
$$MSE = \frac{1}{n}\sum_{i-1}^{n}(y_p - y_a)^2 \tag{2.22}$$

เมื่อ MSE คือ ค่า Mean Square Error

n คือ จำนวนข้อมูลนำเข้า

y_p คือ ค่าข้อมูลส่งออกที่ต้องการ

y_a คือ ค่าข้อมูลส่งออกจริง



รูปที่ 2.26 กราฟ Gradient Descent [17]

1) Stochastic Gradient Descent (SGD)

SGD เป็นอัลกอริทึมที่อัปเดตค่าพารามิเตอร์ในทุก ๆ ชุดข้อมูลฝึกฝน ซึ่งจะอัปเดตแค่ครั้งเดียวต่อการฝึกฝน 1 รอบ ดังสมการที่ (2.23)

$$\theta = \theta - \eta^{\nabla J(\theta;x(i);y(i))} \tag{2.23}$$

เมื่อ $x(i)$ และ $y(i)$ คือ ข้อมูลชุดฝึกฝน

∇J คือ Gradient Vector ของ Cost Function

η คือ อัตราการเรียนรู้

θ คือ ค่าน้ำหนัก

2) Mini Batch Gradient Descent

Mini Batch Gradient Descent เป็นอัลกอริทึมที่ถูกพัฒนาขึ้นจากข้อดีของ Gradient Descent และ SGD โดยอัลกอริทึมนี้จะอัปเดตค่าเป็นชุด

2.6.8.2 Momentum

Momentum เป็นอัลกอริทึมที่ถูกพัฒนาขึ้นเพื่อเร่งความเร็วของ SGD โดยให้ความสำคัญในการพุ่งไปยังทิศทางที่ใกล้จุดกลางมากที่สุด แล้วทำให้ทิศทางที่ไม่เกี่ยวข้องมีความสำคัญลดลง ดังสมการที่ (2.24)

$$V(t) = \gamma V(t-1) + \eta \nabla J(\theta) \tag{2.24}$$

เมื่อ $v(t)$ คือ ค่าของฟังก์ชัน ณ เวลาปัจจุบัน

θ คือ ค่าน้ำหนักซึ่ง $\theta = \theta - V(t)$

η คือ อัตราการเรียนรู้

γ คือ เป็นค่าคงที่ นิยมใช้กันที่ 0.9

2.6.8.3 Adagrad

Adagrad เป็น Optimizer ที่สามารถปรับค่าอัตราการเรียนรู้ (Learning Rate) ให้เหมาะสมกับพารามิเตอร์ได้โดยจะมีการอัปเดตจำนวนมากสำหรับค่าพารามิเตอร์ที่มีจำนวนน้อย และอัปเดตไม่มากสำหรับพารามิเตอร์ที่มีจำนวนมาก จึงเหมาะสมสำหรับข้อมูลที่มีความกระจายสูง ดังสมการที่ (2.25)

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} \cdot g_{t,i} \tag{2.25}$$

เมื่อ $g_{t,i} = \nabla_\theta J(\theta_i)$ ซึ่ง $\nabla_\theta J$ คือ Gradient Vector ของ Cost Function

$\theta_{t+1,i}$ คือ ค่าน้ำหนัก ณ เวลาปัจจุบัน

ε คือ ค่าคงที่ซึ่งมากกว่า 0 และมีค่าน้อยมาก

$\theta_{t,i}$ คือ อัตราการเรียนรู้

$G_{t,ii}$ คือ ผลรวมของ Gradient ถึงเวลา t

2.6.8.4 AdaDelta

AdaDelta เป็น Optimizer ที่พัฒนาขึ้นจาก AdaGrad เพื่อลดปัญหา decaying ของอัตราการเรียนรู้โดยการจำกัดการสะสมค่าการคำนวณของ Gradient เพื่อแก้ค่าน้ำหนักที่เกิดขึ้น ด้วยการหาผลรวมของ Gradient แทนการเก็บค่าน้ำหนักที่ได้รับมาก่อนหน้าซึ่งยังใช้การไม่ได้ ดังสมการที่ (2.26)

$$\nabla\theta_t = -\frac{\eta}{\sqrt{E(g^2)_t + \varepsilon}} g_t \tag{2.26}$$

เมื่อ $\nabla\theta_t$ คือ Gradient Vector ของ Cost Function

g_t คือ ค่า Gradient ณ เวลาปัจจุบัน

η คือ อัตราการเรียนรู้

ε คือ ค่าคงที่ซึ่งมากกว่า 0 และมีค่าน้อยมาก

$E(g^2)_t$ คือ ค่าเฉลี่ยของ Gradient เก่ายกกำลังสอง

2.6.8.5 Adaptive Moment Estimation (Adam)

Adaptive Moment Estimation (Adam) เป็น Optimizer ที่สามารถปรับอัตราการเรียนรู้สำหรับพารามิเตอร์ในแต่ละครั้ง และยังสามารถแก้ปัญหา decaying ของ gradients ในแต่ละขั้นที่ผ่านมาได้เหมือนกับ AdaDelta อีกทั้งยังอธิบายการเกิด decaying average ของ gradients M(t) เหมือนกับ Momentum [18]

2.6.9 Loss Function

Loss Function เป็นฟังก์ชันที่ใช้ชี้วัดประสิทธิภาพ [19] ซึ่งได้ศึกษา 3 ฟังก์ชัน ได้แก่

2.6.9.1 Categorical Cross Entropy Loss Function

Categorical Cross Entropy Loss Function เป็น Loss Function ที่นิยมใช้สำหรับ AI แบบการจำแนกประเภท โดยทำการเปรียบเทียบความผิดพลาดด้วยฟังก์ชัน Softmax

2.6.9.2 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) เป็น Loss Function ที่เปรียบเทียบความผิดพลาด โดยการหาค่าเฉลี่ยของความผิดพลาดจากข้อมูลนำเข้าเป็นคู่ ๆ

2.6.9.3 Mean Squared Error (MSE)

Mean Squared Error (MSE) เป็น Loss Function ที่เปรียบเทียบความผิดพลาด โดยการหาค่าเฉลี่ยของความผิดพลาดยกกำลังสอง

2.6.10 Regularization

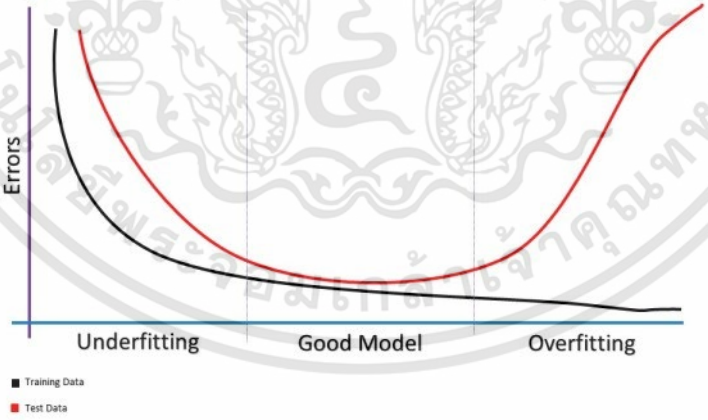
คือการปรับแต่งให้แบบจำลองมีประสิทธิภาพในการทำนายที่ดี ลดความผิดพลาดจากข้อมูลที่ไม่เคยมีมาก่อน ดังนั้นจึงกล่าวอีกอย่างหนึ่งว่า Regularization คือวิธีที่ใช้เพื่อแก้ปัญหา Underfitting หรือ Overfitting ของแบบจำลอง [20]

2.6.10.1 Overfitting

Overfitting เป็นปัญหาเมื่อทำนายด้วยการเรียนรู้เชิงลึก แล้วได้ผลที่ไม่เหมาะสมกับข้อมูลชุดอื่น ๆ ที่ไม่เคยใช้ฝึกฝนเนื่องจากข้อมูลตอบสนองกับข้อมูลรบกวนมากเกินไป แบบจำลองนี้จะมีค่าความแปรปรวนของข้อมูลสูง

2.6.10.2 Underfitting

Underfitting เป็นปัญหาเมื่อทำนายด้วยการเรียนรู้เชิงลึก แล้วได้ผลที่ไม่เหมาะสมกับชุดข้อมูลที่เอาไว้ฝึกฝน อาจเกิดจากข้อมูลมีจำนวนน้อยเกินไปหรือแบบจำลองไม่เหมาะสม โดยกราฟแสดงตัวอย่างการ Overfitting และ Underfitting ดังรูปที่ 2.27



รูปที่ 2.27 กราฟ Overfitting และ Underfitting [22]

2.6.11 Batch Normalization

Batch Normalization คือเทคนิคที่ใช้ระหว่างการฝึกฝน เพื่อปรับเลื่อนให้การกระตุ้นที่อยู่ภายใน Hidden Layer ของโครงข่ายเซลล์ประสาทให้มีขนาดเหมาะสม โดยดูเทียบจากค่าเฉลี่ย

และส่วนเบี่ยงเบนมาตราฐานของทุกการกระตุ้นในแต่ละชั้น ซึ่งอาจใช้ร่วมกับการทำ Dropout การทำ Dropout คือการสุ่มถอดบางโหนดออกในระหว่างการฝึกฝนเพื่อประหยัดเวลาและทรัพยากร Dropout ถือเป็นวิธี Regularization แบบหนึ่งที่ช่วยลดการ Overfitting ของแบบจำลอง [21][22]

2.7 แบบจำลองของโครงข่ายประสาทเทียม

ประเภทของข้อมูลส่งออกของโครงข่ายประสาทเทียมสำหรับการเรียนรู้เชิงลึกมักจะถูกแบ่งเป็น 2 ประเภทดังนี้

2.7.1 การจำแนกประเภท (Classification)

เป็นแบบจำลองที่ใช้ในการทำนายลักษณะข้อมูลที่เป็นกลุ่มและต้องการข้อมูลส่งออกที่ไม่มีความต่อเนื่อง เช่น การตัดสินใจถูกหรือผิด การแยกแยะประเภทสิ่งของ เป็นต้น โดยจะแบ่งเป็น 4 ประเภทคือ

2.7.1.1 การจำแนกแบบไบนารี (Binary Classification)

การจำแนกแบบไบนารีคือการแยกประเภทโดยตัวแปรที่แบ่งเป็นเพียงสองหมวดหมู่ เช่น ผลลัพธ์แบบใช่หรือไม่ใช่ โดยผลลัพธ์มี 2 หมวดหมู่

2.7.1.2 การจำแนกประเภทหลายคลาส (Multi-Class Classification)

การจำแนกประเภทหลายคลาสต่างกับการจำแนกแบบไบนารีตรงที่มีหมวดหมู่มากกว่าสอง ยกตัวอย่างเช่น การจำแนกคำศัพท์ที่คาดว่าจะพิมพ์ในคีย์บอร์ดคือหมวดหมู่เพื่อค้นหารูปภาพที่คล้ายคลึงกับรูปภาพที่อัปโหลด โดยผลลัพธ์ที่มีได้มากกว่า 2 หมวดหมู่

2.7.1.3 การจำแนกประเภทหลายฉลาก (Multi-Label Classification)

การจำแนกประเภทหลายฉลากจะคล้ายกับการจำแนกประเภทหลายคลาสเพื่อเปรียบเทียบให้เข้าใจง่ายขึ้น ยกตัวอย่างเช่น รูปภาพรูปหนึ่งสามารถมีรูปดอกไม้ ท้องฟ้า ก้อนเมฆได้ แต่รูปภาพรูปนั้นจะจัดว่าเป็นหมวดหมู่รูปวาด รูปถ่าย หรือรูปเสีย การจำแนกประเภทหลายฉลากคือติดฉลากว่าในรูปนั้น ๆ มีดอกไม้หรือไม่ มีก้อนเมฆหรือไม่ ส่วนการจำแนกประเภทหลายคลาสจะจำแนกว่ารูปนั้นเป็นรูปวาด รูปถ่าย หรือรูปเสีย

2.7.1.4 การจำแนกแบบข้อมูลไม่เท่าเทียม (Imbalanced Classification)

เป็นปัญหาที่เกิดจากข้อมูลที่ไม่เท่าเทียม (Imbalanced dataset) ตัวอย่างเช่นข้อมูลการทุจริต โดยข้อมูลส่วนใหญ่ย่อมเป็นข้อมูลที่จัดว่าไม่ทุจริตและจะมีเปอร์เซ็นต์น้อยที่จัดว่าเป็นทุจริต เป็นต้น [23]

2.7.2 การถดถอย (Regression)

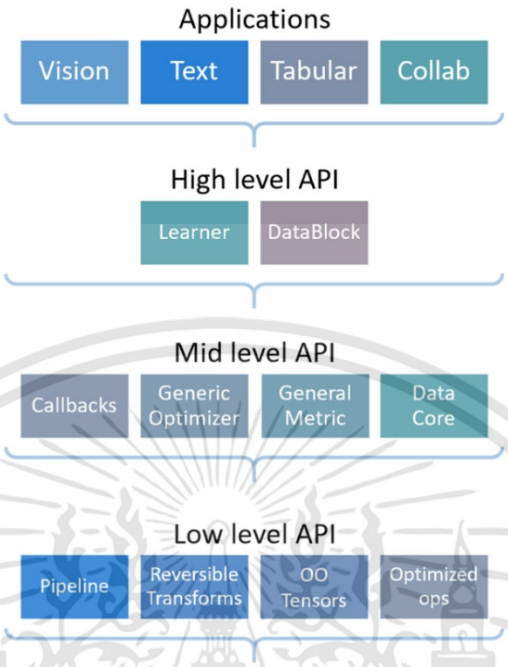
เป็นแบบจำลองที่ใช้ในการทำนายข้อมูลที่ต้องการข้อมูลส่งออกที่มีความต่อเนื่องซึ่งอาศัยข้อมูลในอดีตด้วยตัวอย่างเช่น การทำนายอุณหภูมิในวันถัดไป การทำนายสภาพการจราจรในวันถัดไป เป็นต้น

2.8 Scikit-learn

Scikit-learn เป็นโมดูลหนึ่งของภาษา Python เป็นแพ็คเกจที่รวบรวมไลบรารี (Library) ด้านการเรียนรู้ของเครื่องเอาไว้ และถูกออกแบบมาให้ทำงานร่วมกับไลบรารีของภาษา Python อย่าง Numpy และ Scipy ได้ดี นอกจากนี้ Scikit-learn ยังเป็น Open Source ที่เปิดให้ผู้ที่สนใจสามารถเข้าไปพัฒนาต่อยอดได้ สิ่งที่ทำให้ทุกคนต่างยอมรับคือเป็นแหล่งรวมไลบรารีและอัลกอริทึมที่เน้นไปในด้านของการเรียนรู้ของเครื่อง ซึ่งมีส่วนในการทำ Data Modeling และเป็นเครื่องมือที่แนะนำสำหรับมือใหม่ เพราะเป็น Interface ระดับสูง ทำให้มือใหม่สามารถเข้าใจภาพรวมและขั้นตอนของการเรียนรู้ของเครื่องได้ [24]

2.9 fastai

fastai เป็นไลบรารีที่เขียนด้วยภาษา Python ซึ่งเป็น Open Source พัฒนาโดย Jeremy Howard โดยมีจุดมุ่งหมายคือการทำให้การเรียนรู้เชิงลึกง่ายต่อการนำไปใช้งาน ซึ่ง fastai เป็นส่วนต่อประสานโปรแกรมประยุกต์ (application program interface: API) ระดับสูงที่สร้างขึ้นจาก Pytorch ซึ่งเป็น API ระดับต่ำ โดยโครงสร้างของระดับ API มีดังรูปที่ 2.28 ซึ่งผู้ใช้งานสามารถเขียนด้วย API ระดับสูง โดยที่ไม่ต้องเรียนรู้การใช้งาน API ระดับต่ำด้วยการแบ่งโครงสร้างออกเป็น 2 กลุ่ม คือ Learner สำหรับคำสั่งการเรียนรู้ และ DataBlock สำหรับชุดข้อมูล ทำให้ fastai ใช้งานได้ง่ายและรวดเร็ว



รูปที่ 2.28 ระดับ API

2.10 Flutter Framework

2.10.1 Flutter

Flutter คือ Framework ที่ใช้สร้างส่วนต่อประสานกับผู้ใช้ (user interface: UI) สำหรับโปรแกรมประยุกต์บนโทรศัพท์เคลื่อนที่ โดย Flutter สามารถทำงานข้ามแพลตฟอร์มได้ไม่ว่าจะเป็น iOS หรือ Android ได้ในเวลาเดียวกัน โดยใน Flutter นั้นจะใช้ภาษา dart ในการทำงาน และเพื่อทำให้การออกแบบ UI มีความง่ายและสะดวกยิ่งขึ้น Flutter จึงมี Widget พื้นฐานมาให้หลัก ๆ โดยมีอยู่ 2 ชนิดคือ Stateless Widget และ Stateful Widget โดยที่ Stateless Widget จะใช้สร้าง Widget ที่ไม่มีการจัดการสถานะการทำงานใด ๆ เช่น การแสดงข้อความ ไอคอน (Icon) หรือรูปภาพที่ไม่มีภาพเคลื่อนไหว เข้ามาเกี่ยวข้อง ส่วนการสร้าง Widget ที่มีการจัดการสถานะการทำงานต่าง ๆ เช่น การสร้าง ไอคอน ที่มีการใส่ภาพเคลื่อนไหวให้สามารถขยับไปมาได้ และปุ่มกดต่าง ๆ บนหน้า UI จะใช้ Stateful Widget ในการสร้าง

จุดเด่นที่สำคัญของ Flutter คือ ระบบ Hot Reload ซึ่งจะเข้ามาช่วยในส่วนของการ Reload โดยเมื่อมีการทดสอบ การสร้าง การเพิ่มคุณลักษณะ หรือการกระทำต่าง ๆ กับ UI จะต้องมีการ reload เพื่ออัปเดตหน้า UI โดยระบบนี้จะใช้เวลาในการ Reload เพียงเสี้ยววินาที ทำให้การ

พัฒนา UI ของโปรแกรมประยุกต์มีความรวดเร็วขึ้นอย่างมาก และยังมีจุดเด่นอื่น ๆ ไม่ว่าจะเป็น Build-In ที่ช่วยในการออกแบบ UI ให้มีความสวยงามยิ่งขึ้นอย่างเช่น Material Design และ Cupertino (iOS-flavor) มี Framework ที่ช่วยให้การทำภาพเคลื่อนไหวต่าง ๆ หรือ Gesture ของ UI เป็นเรื่องที่ไม่ยาก และยังสามารถใช้งานร่วมกับสิ่งแวดล้อมสำหรับการพัฒนาแบบเบ็ดเสร็จ (integrated development environment: IDE) ที่กำลังเป็นที่นิยมอยู่ในปัจจุบันอย่าง VS Code และ Android Studio ซึ่งจุดเด่นเหล่านี้ทำให้การพัฒนาเป็นไปได้ง่ายขึ้น [25]

2.10.2 ภาษา Dart

Dart เป็นภาษาโปรแกรมที่เอาไว้สำหรับสร้างโปรแกรมประยุกต์บนแพลตฟอร์มที่หลากหลาย และสร้างได้ทั้ง Mobile Desktop Server และ Web สิ่งที่เป็นที่นิยมที่สุดในภาษา Dart คือนำไปใช้ร่วมกับ Flutter ที่เป็นเครื่องมือช่วยสร้าง UI ของ Google ซึ่งใช้ได้ทั้งกับ Android และ iOS หรือใน Desktop กับ Web ก็ได้ภาษานี้ถูกสร้างโดย Google และปล่อยให้ใช้งานแบบ Open Source ทำให้สามารถนำไปใช้งานได้โดยไม่มีค่าใช้จ่าย และถูกออกแบบมาให้ใช้งานได้ง่าย และมีประสิทธิภาพแบบภาษาเชิงวัตถุอื่น ๆ อย่าง Java, C# และ C++

ภาษา Dart เป็นภาษากลุ่ม Compiler นั่นคือจำต้อง Compile ก่อนนำโปรแกรมไปดำเนินงาน ไม่เหมือนภาษากลุ่ม Script ที่ใช้ Interpreter ในการดำเนินงาน Source code ตรง ๆ โครงสร้างของภาษา Dart คล้ายกับ C/C++ และ Java โดยที่จะมีความเป็นภาษาแบบ Structure Programming แต่ก็ยังมีความสามารถแบบภาษาประเภท Object Oriented Programming ด้วย นั่นคือมี Class และ Inheritance ให้ใช้งาน เป้าหมายของการสร้างภาษา Dart ขึ้นมา คือการสร้างภาษาเชิงโครงสร้างที่ยืดหยุ่นมากพอ (Structured yet flexible language) และเป็นการออกแบบตัวภาษาไปพร้อมกับตัว Engine สำหรับดำเนินงานภาษา เพื่อแก้ปัญหาโปรแกรมทำงานช้าและกิน Memory ซึ่งเป้าหมายของภาษา Dart คือเป็นภาษาที่เรียนรู้ง่าย ทำงานได้บนอุปกรณ์พกพาขนาดเล็กไปจนถึงเซิร์ฟเวอร์ และมีจุดเด่นคือภาษา Dart เป็นภาษาที่ใช้ในการสร้างโปรแกรมประยุกต์ด้วย Flutter Framework ซึ่งปัจจุบันภาษา Dart มีออกมา 2 เวอร์ชันคือ Dart1 และ Dart2 [26]

2.11 ส่วนต่อประสานโปรแกรมประยุกต์ (application program interface: API)

API คือ คำสั่ง (Code) ที่อนุญาตให้โปรแกรมซอฟต์แวร์ (Software Program) สามารถสื่อสารระหว่างกันได้ เป็นช่องทางสำหรับขอใช้บริการคำสั่งจากระบบปฏิบัติการ

(Operation System: OS) หรือโปรแกรมประยุกต์ อื่น ๆ ซึ่งใช้งานโดยติดตั้งฟังก์ชัน และเรียกใช้งานตามเอกสารที่เขียนไว้ โดย API สร้างขึ้นจากส่วนสำคัญ 2 อย่าง คือ

1) ข้อกำหนดที่จะอธิบายการแลกเปลี่ยนข้อมูลระหว่างโปรแกรม ซึ่งทำออกมาในลักษณะเอกสาร เพื่อบอกว่า Request/Response ต้องเป็นอย่างไร

2) ซอฟต์แวร์ (Software) ที่เขียนขึ้นตามข้อกำหนด และทำการเผยแพร่ออกไปให้ใช้งาน โดยปกติโปรแกรมประยุกต์ที่มี APIs จะต้องถูกเขียนเป็นภาษา Programming และพัฒนาเพิ่มได้ง่าย จึงจำเป็นต้องมีการตรวจสอบโครงสร้าง API ดังนั้น API ที่ดี ผู้ที่ออกแบบต้องให้ความสำคัญในการทดสอบ เพื่อตรวจสอบตรรกะ (Logic) ที่สามารถเกิดขึ้นได้จากการใช้งาน [28]

2.12 FastAPI Framework

FastAPI เป็น Framework สำหรับ API ที่มีประสิทธิภาพสูง เรียนรู้ได้ง่าย เขียนได้เร็ว และพร้อมสำหรับผลิตภัณฑ์จริง ซึ่ง FastAPI สามารถสร้าง API ได้ด้วยภาษา Python โดยถูกออกแบบมาให้ง่ายต่อการพัฒนา สามารถที่จะสร้าง API ขึ้นมาได้อย่างรวดเร็ว อีกทั้งยังมีประสิทธิภาพการทำงานเร็วเทียบเท่า NodeJS และ Go ซึ่ง FastAPI นั้นรองรับการทำงานแบบ Asynchronous และมี Uvicorn เป็นตัวดำเนินงานเซิร์ฟเวอร์ [29]

FastAPI มีคุณสมบัติน่าสนใจอย่างมากดังนี้

- เป็น MIT License
- รองรับการสร้าง API หลากหลายรูปแบบ และรองรับ OAuth2
- มีระบบสร้างเอกสาร API แบบอัตโนมัติ ที่สามารถทดลอง API ได้ผ่านหน้าเว็บเอกสาร
- API เข้ากันได้กับ OpenAPI และ JSON Schema

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ

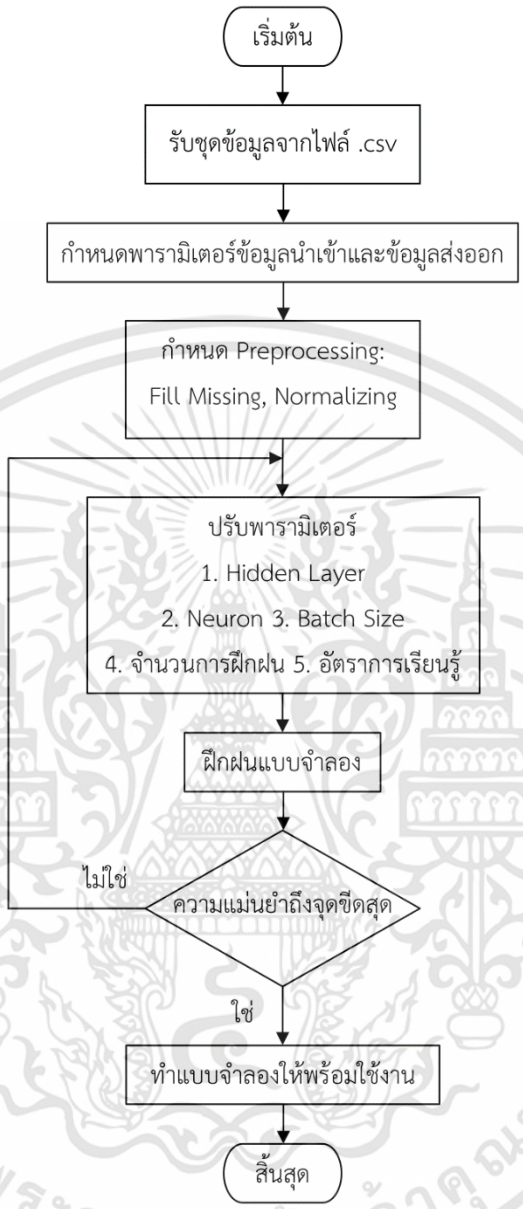
3.1.1 การออกแบบการทำงานของระบบ

การทำงานของระบบจะแบ่งเป็น 3 ส่วน คือ ส่วนนำเข้าข้อมูล ส่วนประมวลผลข้อมูล และส่วนส่งออกข้อมูล โดยจะรับค่า RSSI จากส่วนนำเข้าข้อมูล ส่งต่อไปยังส่วนประมวลผลเพื่อนำมาวิเคราะห์และทำนายตำแหน่งภายในอาคาร แล้วจึงส่งพิกัดของตำแหน่งที่ได้จากส่วนประมวลผลไปยังโปรแกรมประยุกต์บนอุปกรณ์เคลื่อนที่ดังรูปที่ 3.1

รูปที่ 3.1 บล็อกไดอะแกรมของระบบ

3.1.2 การออกแบบระบบการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

โครงข่ายเซลล์ประสาทนั้นมีข้อดีคือ ให้ความแม่นยำ ถูกต้อง และเชื่อถือได้ สามารถใช้ทำนายผลกับข้อมูลที่ไม่เป็นเชิงเส้น (Non-Linear) ได้ แต่จำเป็นที่จะต้องมีชุดข้อมูลขนาดใหญ่ มีขั้นตอนที่ค่อนข้างซับซ้อน และมีตัวแปรอิสระที่ไม่ทราบว่ามีผลกับตัวแปรตามมากน้อยอย่างไร จึงได้ออกแบบขั้นตอนการสร้างแบบจำลอง โดยการปรับพารามิเตอร์เพื่อหาจุดขีดสุดของความแม่นยำที่ระบบสามารถทำได้ ซึ่งมีผังงานดังรูปที่ 3.2 โดยชุดข้อมูลที่จะนำมาฝึกฝนต้องประกอบไปด้วยข้อมูลนำเข้าที่จะป้อนให้กับระบบ และข้อมูลส่งออกที่เป็นผลลัพธ์ของการทำนายนำมาใช้ในการฝึกฝน เนื่องจากค่าของข้อมูลแต่ละข้อมูลนำเข้าและข้อมูลส่งออกมีช่วงที่กว้างและมีหลายระดับ จึงจำเป็นต้องแปลงให้อยู่ในมาตราส่วนที่เหมือนกันคือ มีค่าเฉลี่ยเท่ากับ 0 มีความแปรปรวนเท่ากับ 1 โดยวิธีการ Normalize ซึ่งเป็นกระบวนการของ Preprocessing เมื่อทำการปรับพารามิเตอร์และการฝึกฝนแบบจำลองเสร็จสิ้น จะต้องมีการประเมินความแม่นยำของแบบจำลอง และทำซ้ำจนกว่าจะพบจุดขีดสุดของระบบที่ทำได้ จากนั้นจึงนำแบบจำลองมาใช้งานต่อไป



รูปที่ 3.2 ผังงานการออกแบบระบบการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

3.1.3 การออกแบบแบบจำลองการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

ในการออกแบบระบบการระบุตำแหน่ง มีปัจจัยสำคัญคือเครื่องรับและเครื่องส่ง ซึ่งจำเป็นที่จะต้องมีการออกแบบการติดตั้ง และมีขั้นตอนที่ยุ่งยากต่อการกำหนดสภาพแวดล้อม ขนาดพื้นที่ และปัจจัยอื่น ๆ จึงได้ทำการออกแบบแบบจำลอง เพื่อง่ายต่อการปรับพารามิเตอร์ การเก็บข้อมูลปริมาณมาก และการประเมินแบบจำลอง โดยจะมีการจำลองชุดข้อมูลด้วยการจำลองพื้นที่ตำแหน่งเครื่องรับ-ส่ง และสัญญาณรบกวน ดังรูปที่ 3.3

รูปที่ 3.3 ผังงานแบบจำลองชุดข้อมูล RSSI

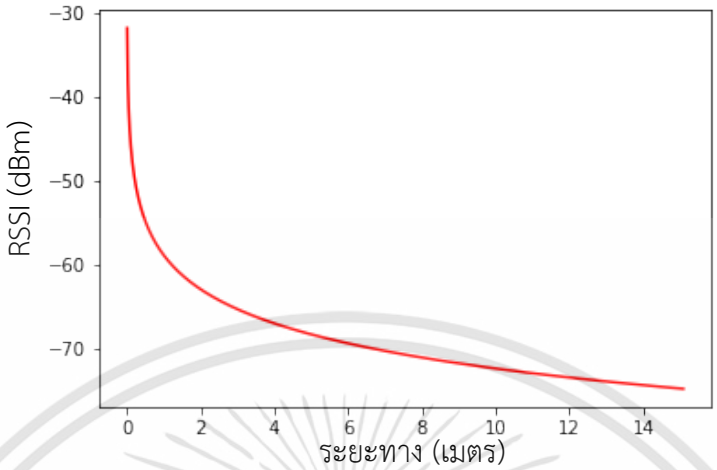
3.1.3.1 การประมาณระยะทางจากค่า RSSI

เนื่องจากสถานการณ์ที่ทำให้ไม่สามารถติดตั้งอุปกรณ์ในสถานที่เพื่อทำการเก็บตัวอย่างข้อมูลตัวอย่างไปใช้ในการคำนวณหาพารามิเตอร์ของแบบจำลอง Path Loss ได้ จึงได้จำลองแบบจำลองการแพร่กระจายคลื่นวิทยุ โดยใช้ข้อมูลจากเอกสารอ้างอิง [30] มาทำการประมาณค่าพารามิเตอร์ ซึ่งข้อมูลที่นำมาใช้ ได้แก่ ค่าเฉลี่ยของ RSSI ที่ระยะทาง 1 2 3 4 5 7.5 10 และ 15 เมตร โดยกำหนดให้ RSSI ที่ระยะ 1 เมตร เป็นค่าอ้างอิง ($RSSI_{d_0}$) แล้วทำการคำนวณหาค่าเฉลี่ยของ η จากสมการที่ (2.6) ให้ผลลัพธ์ดังตารางที่ 3.1 ซึ่งค่าเฉลี่ยของ η เท่ากับ 1.3561 จะได้ค่าพารามิเตอร์สำหรับการนำไปจำลองแบบจำลองการแพร่กระจายคลื่นวิทยุดังนี้คือ $RSSI_{d_0}$ เท่ากับ -58.889 dBm d_0 เท่ากับ 1 เมตร และ η เท่ากับ 1.3561

ตารางที่ 3.1 ผลการคำนวณหาค่าของ η จากค่า RSSI และระยะทาง

ระยะทาง (เมตร)	RSSI (dBm)	η
1	-58.889	-
2	-60.732	0.6122
3	-66.086	1.5084
4	-66.577	1.2769
5	-69.6	1.5324
7.5	-69.474	1.2096
10	-76.439	1.7550
15	-77.685	1.5982

เมื่อนำค่าพารามิเตอร์ที่ได้มาประกอบในสมการของแบบจำลอง Path Loss แล้วทำการวาดกราฟความสัมพันธ์ระหว่างระยะทางและค่า RSSI จะได้ความสัมพันธ์ดังรูปที่ 3.4



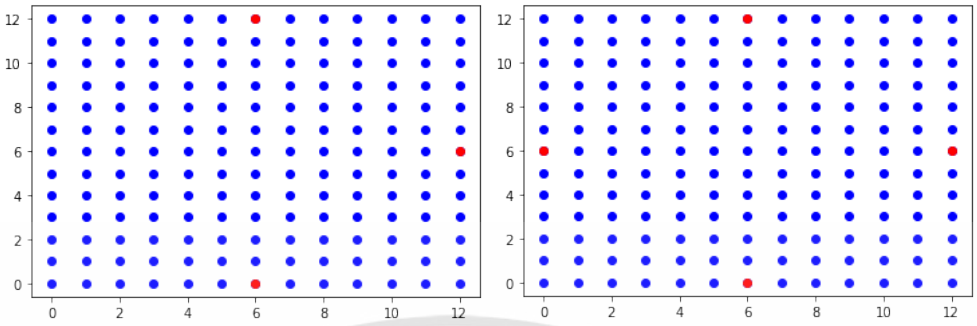
รูปที่ 3.4 กราฟความสัมพันธ์ระหว่างระยะทางกับค่า RSSI จากการหาค่าพารามิเตอร์

ค่าพารามิเตอร์ที่ได้จะถูกนำไปใช้ในการสร้างแบบจำลองชุดข้อมูล ซึ่งอาศัยความสัมพันธ์ของค่า RSSI กับระยะทางในการประมาณค่าผลลัพธ์แทนการเก็บข้อมูลในเบื้องต้นเท่านั้น โดยหลังจากประเมินความสามารถของการระบุตำแหน่งจากแบบจำลองนี้ จะมีการทดลองเก็บข้อมูลในทางปฏิบัติต่อไป

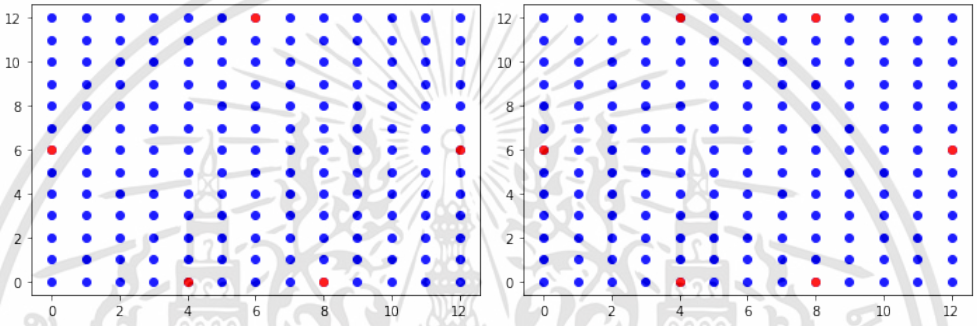
3.1.3.2 แบบจำลองชุดข้อมูล

1) แบบจำลองพื้นที่ภายในอาคารและตำแหน่งอุปกรณ์ติดตั้ง

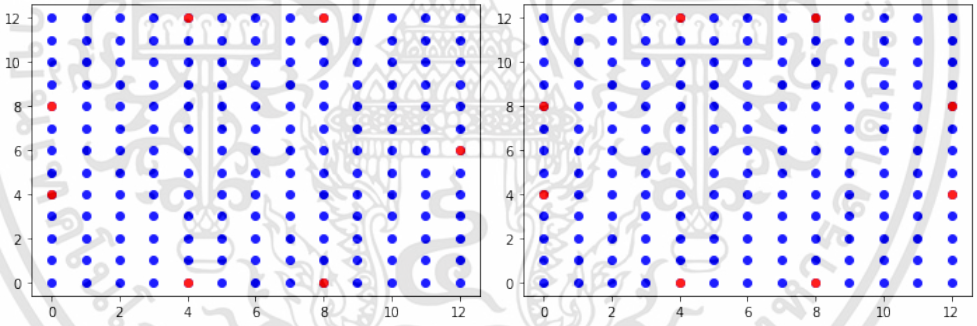
เนื่องจากการใช้งานโครงข่ายเซลล์ประสาท เป็นการทำนายผลลัพธ์แบบการจำแนกประเภท ซึ่งให้ผลลัพธ์เป็นการระบุ class ของผลลัพธ์ที่เป็นไปได้ จึงได้แทนตำแหน่งแต่ละพิกัดด้วย class ที่แตกต่างกัน พื้นที่จำลองถูกกำหนดด้วยขนาด กว้าง x ยาว (หน่วย) ในระนาบ 2 มิติ ซึ่งจะถูกคำนวณเป็นระยะทางจริงในหน่วยเมตรด้วยอัตราส่วนที่ถูกต้อง ตำแหน่งแต่ละจุดในพื้นที่ถูกกำหนดด้วยระบบพิกัด (x, y) ในระยะห่างเท่า ๆ กันทุกตำแหน่งซึ่งเป็นตำแหน่งที่เป็นไปได้ที่จะเป็นตำแหน่งเครื่องรับ และตำแหน่งเครื่องส่งจะระบุด้วยระบบพิกัดเช่นเดียวกัน ตัวอย่างเช่น การจำลองตำแหน่งในพื้นที่ขนาด 4.8 x 4.8 เมตร เป็น 12 x 12 หน่วย ด้วยอัตราส่วน 0.4 เมตรต่อ 1 หน่วย โดยจุดสีน้ำเงินแทนตำแหน่งเครื่องรับ และจุดสีแดงแทนตำแหน่งเครื่องส่งจำนวน 3 – 12 ตัว ดังรูปที่ 3.5 โดยมีจำนวนเครื่องส่ง (ก) 3 ตัว (ข) 4 ตัว (ค) 5 ตัว (ง) 6 ตัว (จ) 7 ตัว (ฉ) 8 ตัว (ช) 9 ตัว (ซ) 10 ตัว (ฌ) 11 ตัว (ญ) 12 ตัว



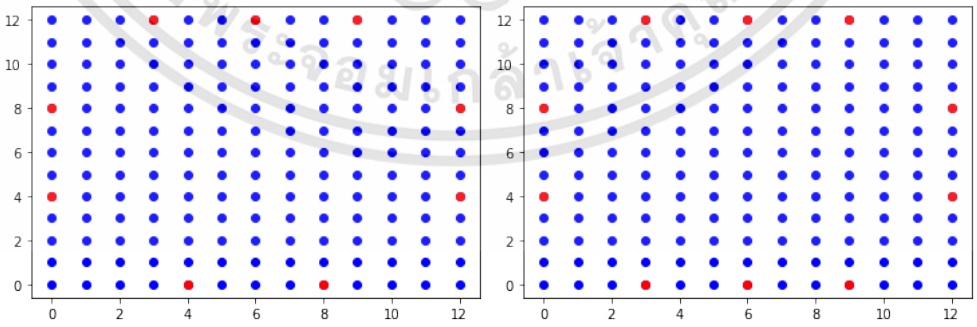
(ก) (ข)



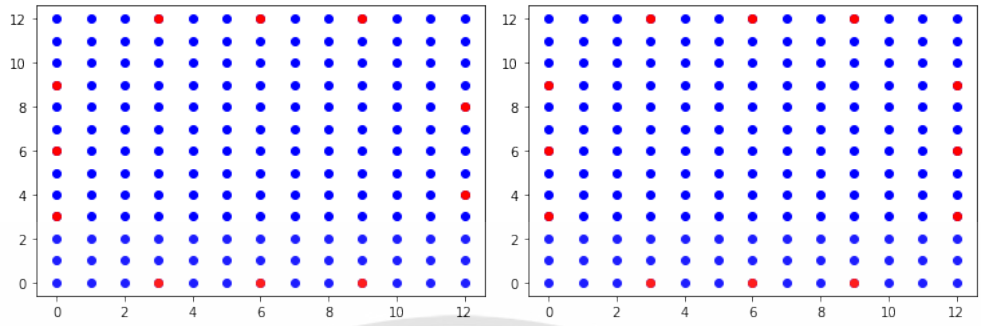
(ข) (ง)



(จ) (ฉ)



(ช) (ซ)



(ฌ) (ญ)

รูปที่ 3.5 ตัวอย่างการจำลองตำแหน่งในพื้นที่ขนาด 12 x 12 หน่วย

2) แบบจำลองข้อมูล RSSI

ตำแหน่งพิกัดแต่ละตำแหน่งจะถูกคำนวณหาระยะห่างจากตำแหน่งเครื่องส่งด้วยสมการที่ (3.1) แล้วคูณด้วยอัตราส่วนที่กำหนดเพื่อให้ได้ระยะจริงในหน่วยเมตร จะได้ผลลัพธ์เป็นระยะห่างจากตำแหน่งเครื่องส่ง n ตัว

$$d_n = \sqrt{\left(x_{rx} - x_{tx}\right)^2 + \left(y_{rx} - y_{tx}\right)^2} \tag{3.1}$$

เมื่อได้ระยะห่างแล้วจะถูกนำไปคำนวณด้วยสมการที่ (2.6) ด้วยพารามิเตอร์ที่ได้จากการคำนวณในข้อ 3.1.3.1 ให้ผลลัพธ์เป็นค่า RSSI จากเครื่องส่งทั้งหมดไปยังตำแหน่งพิกัดทุกตำแหน่ง โดยเพิ่มเงื่อนไขให้ระยะที่ต่ำกว่า 0.1 เมตรจะถูกกำหนดให้มี RSSI เท่ากับที่ระยะ 0.1 เมตร ตัวอย่างข้อมูลจากการคำนวณแสดงดังรูปที่ 3.6 เมื่อค่าในคอลัมน์ rssi_0 rssi_1 rssi_2 rssi_3 rssi_4 rssi_5 rssi_6 และ rssi_7 คือ RSSI จากเครื่องส่งแต่ละตัวที่เครื่องรับในตำแหน่งพิกัดนั้น ๆ รับค่าได้จากเครื่องส่งทั้ง 8 ตัว และค่าในคอลัมน์ position คือหมายเลขตำแหน่งแทนพิกัด ซึ่งมีค่าพิกัดจะอยู่ในคอลัมน์ x และ y

	A	B	C	D	E	F	G	H	I	J	K	L
1	rssi_0	rssi_1	rssi_2	rssi_3	rssi_4	rssi_5	rssi_6	rssi_7	position	x	y	
2	-61.6571	-65.7393	-68.4376	-69.2102	-69.2102	-68.4376	-65.7393	-61.6571	0	0	0	
3	-59.9628	-64.9529	-67.9806	-68.8651	-68.9898	-68.3058	-65.785	-61.8356	1	0	1	
4	-57.5748	-64.045	-67.4906	-68.5103	-68.7844	-68.208	-65.9179	-62.3142	2	0	2	
5	-53.4925	-62.9713	-66.9638	-68.1477	-68.5987	-68.1477	-66.1268	-62.9713	3	0	3	
6	-45.328	-61.6571	-66.3964	-67.7805	-68.4376	-68.1273	-66.3964	-63.6982	4	0	4	
7	-53.4925	-59.9628	-65.785	-67.4134	-68.3058	-68.1477	-66.7104	-64.428	5	0	5	
8	-57.5748	-57.5748	-65.1279	-67.0535	-68.208	-68.208	-67.0535	-65.1279	6	0	6	
9	-59.9628	-53.4925	-64.428	-66.7104	-68.1477	-68.3058	-67.4134	-65.785	7	0	7	
10	-61.6571	-45.328	-63.6982	-66.3964	-68.1273	-68.4376	-67.7805	-66.3964	8	0	8	
11	-62.9713	-53.4925	-62.9713	-66.1268	-68.1477	-68.5987	-68.1477	-66.9638	9	0	9	
12	-64.045	-57.5748	-62.3142	-65.9179	-68.208	-68.7844	-68.5103	-67.4906	10	0	10	
13	-64.9529	-59.9628	-61.8356	-65.785	-68.3058	-68.9898	-68.8651	-67.9806	11	0	11	
14	-65.7393	-61.6571	-61.6571	-65.7393	-68.4376	-69.2102	-69.2102	-68.4376	12	0	12	
15	-61.8356	-65.785	-68.3058	-68.9898	-68.8651	-67.9806	-64.9529	-59.9628	13	1	0	
16	-60.273	-65.0124	-67.8261	-68.6161	-68.6161	-67.8261	-65.0124	-60.273	14	1	1	
17	-58.2319	-64.1257	-67.3073	-68.2278	-68.3818	-67.7106	-65.184	-61.0456	15	1	2	
18	-55.5337	-63.0868	-66.7433	-67.8261	-68.1679	-67.6391	-65.4495	-62.0039	16	1	3	
19	-53.4925	-61.8356	-66.1268	-67.4134	-67.9806	-67.6149	-65.785	-62.9713	17	1	4	
20	-55.5337	-60.273	-65.4495	-66.994	-67.8261	-67.6391	-66.1669	-63.8767	18	1	5	
21	-58.2319	-58.2319	-64.7021	-66.575	-67.7106	-67.7106	-66.575	-64.7021	19	1	6	
22	-60.273	-55.5337	-63.8767	-66.1669	-67.6391	-67.8261	-66.994	-65.4495	20	1	7	
23	-61.8356	-53.4925	-62.9713	-65.785	-67.6149	-67.9806	-67.4134	-66.1268	21	1	8	
24	-63.0868	-55.5337	-62.0039	-65.4495	-67.6391	-68.1679	-67.8261	-66.7433	22	1	9	
25	-64.1257	-58.2319	-61.0456	-65.184	-67.7106	-68.3818	-68.2278	-67.3073	23	1	10	
26	-65.0124	-60.273	-60.273	-65.0124	-67.8261	-68.6161	-68.6161	-67.8261	24	1	11	
27	-65.785	-61.8356	-59.9628	-64.9529	-67.9806	-68.8651	-68.9898	-68.3058	25	1	12	
28	-62.3142	-65.9179	-68.208	-68.7844	-68.5103	-67.4906	-64.045	-57.5748	26	2	0	
29	-61.0456	-65.184	-67.7106	-68.3818	-68.2278	-67.3073	-64.1257	-58.2319	27	2	1	
30	-59.6159	-64.3553	-67.169	-67.959	-67.959	-67.169	-64.3553	-59.6159	28	2	2	

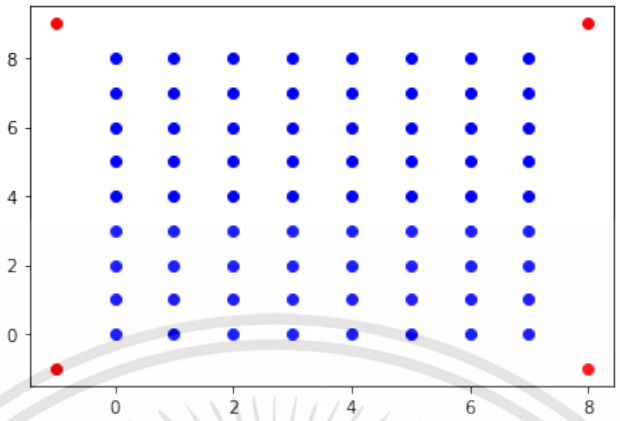
รูปที่ 3.6 ตัวอย่างข้อมูลจำลองค่า RSSI ตำแหน่งที่ 0 ถึง 28 จาก 169 ตำแหน่ง

ข้อมูลจำลอง RSSI นี้จะถูกเพิ่ม Sample ด้วยจำนวนที่กำหนด ซึ่งในการจำลองต่อไปนี้จะใช้จำนวน 50 samples แล้วบวกด้วยค่าที่สุ่มแบบการแจกแจงปกติ (Normal Distribution) โดยกำหนดค่าเฉลี่ยเท่ากับ 0 ค่าความแปรปรวนเท่ากับ 0.5 ซึ่งให้ผลลัพธ์เฉลี่ยไม่เกิน ±3 dBm แล้วนำผลลัพธ์ทุก Sample มาทำการสุ่มสลับลำดับข้อมูลกัน เพื่อนำไปใช้ในการฝึกและทดสอบแบบจำลองต่อไป

3.1.4 การออกแบบการติดตั้งเครื่องส่งสัญญาณและตำแหน่งทดสอบภายในห้องทดลองและทางเดินภายในอาคาร

3.1.4.1 ห้อง Co-working space

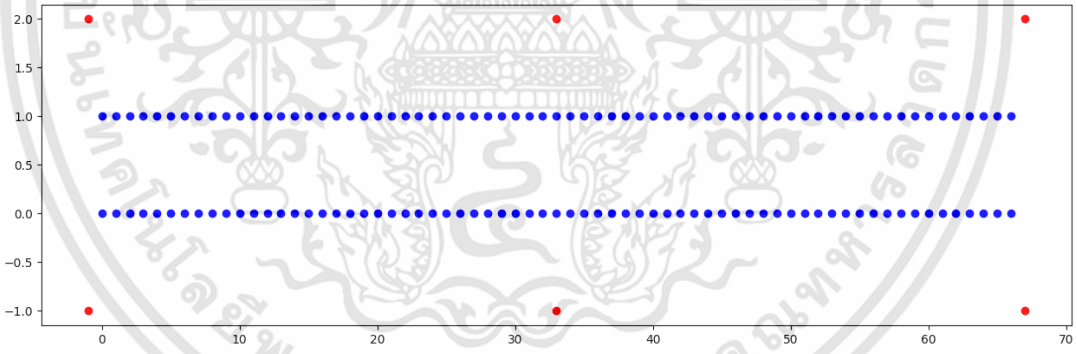
ออกแบบตำแหน่งการติดตั้งเครื่องส่งสัญญาณภายในห้อง Co-working space ตึกภาควิชาโทรคมนาคม ที่มีขนาดกว้าง 3.5 เมตร และยาว 4.0 เมตร โดยภายในห้องทดลองแบ่งตำแหน่งทดสอบเป็นตารางขนาด 0.4 x 0.4 เมตร จำนวน 72 ตำแหน่ง ซึ่งแต่ละตำแหน่งแสดงเป็นจุดสีน้ำเงิน และได้ออกแบบให้มีการติดตั้งเครื่องส่งจำนวน 4 ตัว แสดงเป็นจุดสีแดง ดังรูปที่ 3.7



รูปที่ 3.7 แผนผังแสดงตำแหน่งการติดตั้งเครื่องส่งห้อง Co-working space

3.1.4.2 บริเวณทางเดินตึกภาควิชาโทรคมนาคมชั้น 3

ออกแบบตำแหน่งการติดตั้งเครื่องส่งสัญญาณบริเวณตึกภาควิชาโทรคมนาคมชั้น 3 ที่มีขนาดกว้าง 2.1 เมตร และยาว 33.0 เมตร โดยบริเวณทางเดินแบ่งตำแหน่งทดสอบเป็นตารางขนาด 0.5 x 0.5 เมตร จำนวน 134 ตำแหน่ง ซึ่งแต่ละตำแหน่งแสดงเป็นจุดสีน้ำเงิน และได้ออกแบบให้มีการติดตั้งเครื่องส่งจำนวน 6 ตัว แสดงเป็นจุดสีแดง ดังรูปที่ 3.8

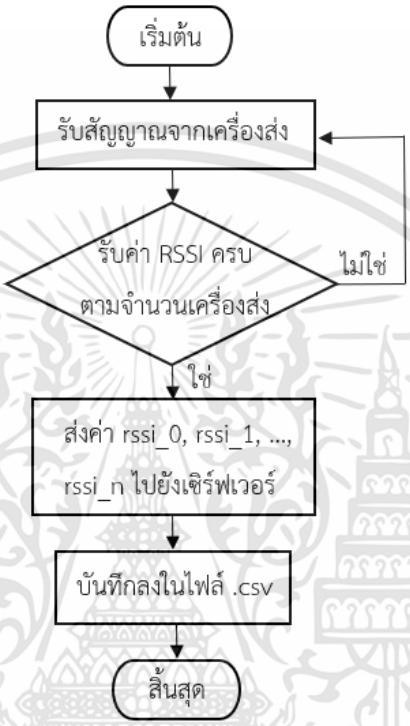


รูปที่ 3.8 แผนผังแสดงตำแหน่งการติดตั้งเครื่องส่งบริเวณทางเดินตึกภาควิชาโทรคมนาคมชั้น 3

3.1.5 การออกแบบการจัดเก็บข้อมูล RSSI

ในการจัดเก็บข้อมูล RSSI นั้น จะทำการออกแบบโปรแกรมประยุกต์ให้ทำการรับค่า RSSI จากเครื่องส่ง จากนั้นให้ทำการส่งค่า RSSI ของเครื่องส่งที่มี mac address ที่กำหนด และตำแหน่งที่ทำการเก็บค่า RSSI ไปยังเซิร์ฟเวอร์ โดยใช้ Hypertext Transfer Protocol (HTTP) เป็นโพรโทคอลที่ใช้ในการสื่อสารผ่านอินเทอร์เน็ต ซึ่งจะกำหนดเงื่อนไขให้ทำการส่งชุดข้อมูลค่า RSSI และตำแหน่งไปยังเซิร์ฟเวอร์ เมื่อมีค่า RSSI ครบตามจำนวนเครื่องส่งที่กำหนด

จากนั้นฝั่งเซิร์ฟเวอร์จะนำข้อมูลที่ได้รับมาบันทึกไว้ในไฟล์ .csv ที่สร้างขึ้น ดังแผนผังการทำงาน รูปที่ 3.9 และหน้าโปรแกรมประยุกต์แสดงดังรูปที่ 3.10

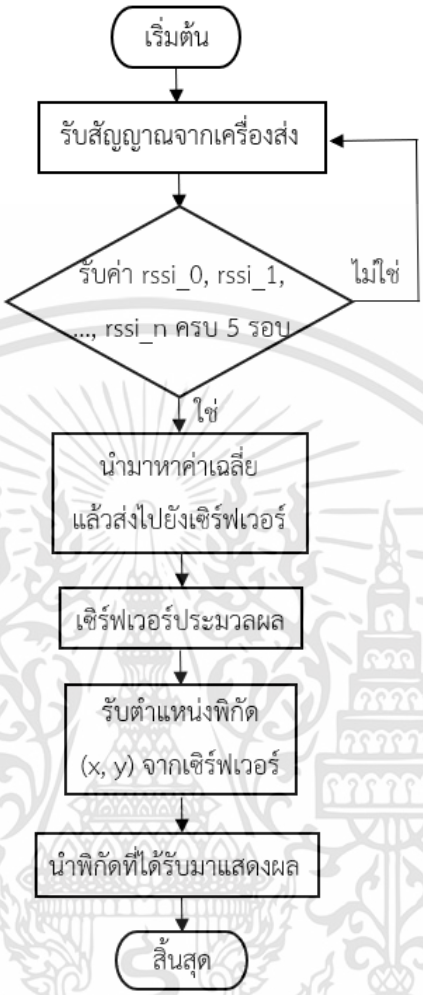


รูปที่ 3.9 แผนผังการทำงานของโปรแกรมประยุกต์ที่ใช้ในการเก็บข้อมูล RSSI

รูปที่ 3.10 โปรแกรมประยุกต์ที่ใช้ในการเก็บข้อมูล RSSI

3.1.6 การออกแบบโปรแกรมประยุกต์ระบุตำแหน่งภายในอาคาร

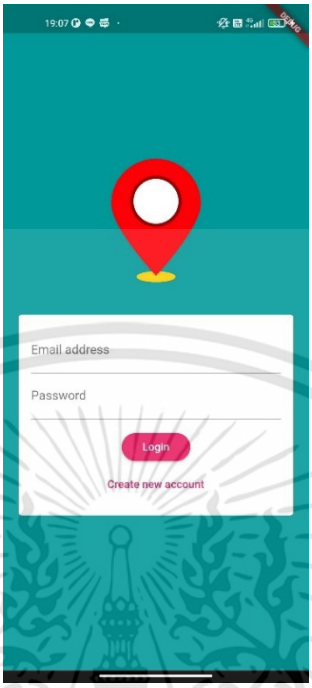
โปรแกรมประยุกต์จะถูกนำมาใช้งานบนโทรศัพท์เคลื่อนที่ ซึ่งทำหน้าที่เป็นเครื่องรับและเป็นส่วนติดต่อกับผู้ใช้งาน โดยในการรับค่า RSSI จะรับผ่านสัญญาณ Wi-Fi จากเครื่องส่งที่ติดอยู่ตามจุดต่างๆ ทั้งหมด 5 รอบจากนั้นนำมาหาค่าเฉลี่ย แล้วนำค่าที่ได้ส่งไปยังเซิร์ฟเวอร์ประมวลผลผ่าน API เมื่อเซิร์ฟเวอร์ประมวลผลเสร็จสิ้น โปรแกรมประยุกต์จะรับข้อมูลตำแหน่งเป็นพิกัด (x, y) จากเซิร์ฟเวอร์มาแสดงผล เพื่อระบุตำแหน่งของผู้ใช้ ซึ่งแผนผังการทำงานของโปรแกรมประยุกต์แสดงดังรูป 3.11



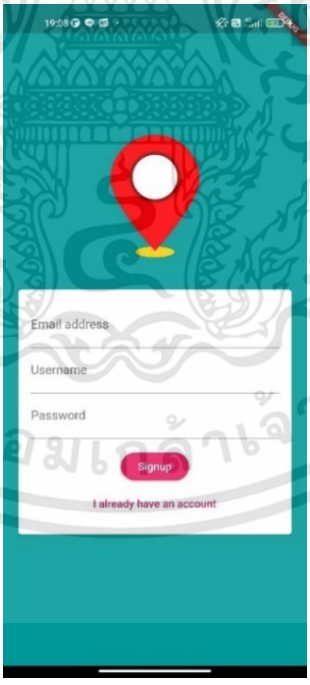
รูปที่ 3.11 แผนผังการทำงานของโปรแกรมประยุกต์

3.1.6.1 ออกแบบระบบยืนยันตัวตนบนโปรแกรมประยุกต์

ผู้จัดทำใช้โปรแกรม Visual Studio Code และภาษา Dart ในการออกแบบและตกแต่งโปรแกรมประยุกต์ โดยโปรแกรมประยุกต์ในส่วนแรก จะเป็นส่วนของการเริ่มต้นใช้งาน ซึ่งแบ่งออกเป็นหน้าเริ่มต้นใช้งานโปรแกรมประยุกต์สำหรับเข้าสู่ระบบเพื่อเข้าใช้งานและหน้าลงทะเบียนผู้ใช้งาน โดยหน้าเริ่มต้นใช้งานโปรแกรมประยุกต์ แสดงดังรูปที่ 3.12 และหน้าลงทะเบียนแสดงดังรูปที่ 3.13

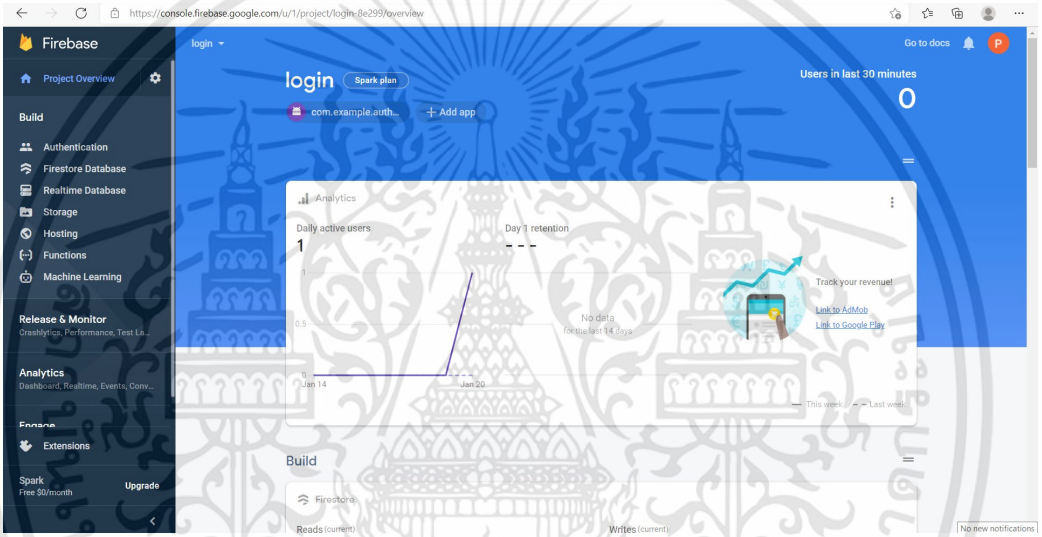


รูปที่ 3.12 หน้าเริ่มต้นใช้งานโปรแกรมประยุกต์

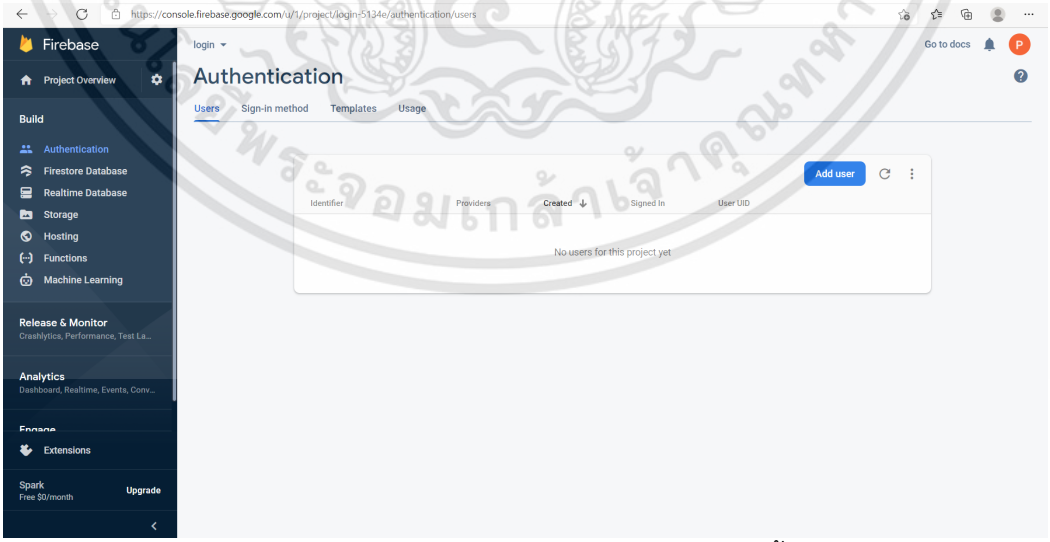


รูปที่ 3.13 หน้าลงทะเบียนผู้ใช้งาน

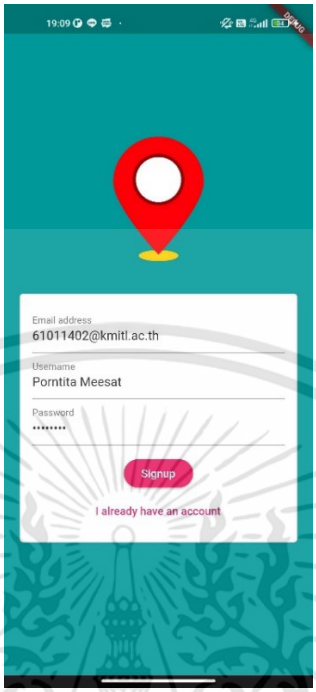
ในส่วนของการออกแบบฐานข้อมูลในส่วนของการยืนยันตัวตนบนโปรแกรมประยุกต์ ผู้จัดทำได้ใช้ Firebase ในการออกแบบฐานข้อมูล โดยเริ่มต้นสร้างโปรเจคที่ Firebase console (https://console.firebase.google.com) ทำการสร้างโปรเจคใหม่ แล้วทำการติดตั้ง Firebase บนโปรแกรมประยุกต์ระบบปฏิบัติการแอนดรอยด์ เมื่อติดตั้งเรียบร้อยแล้วจะแสดงผลดังรูปที่ 3.14 จากนั้นทำการสร้างฐานข้อมูล Authentication สำหรับเก็บข้อมูลผู้ใช้งานแสดงดังรูปที่ 3.15 และเมื่อผู้ใช้งานทำการลงทะเบียนใช้งานโปรแกรมประยุกต์ ดังรูปที่ 3.16 ข้อมูลผู้ใช้งานจะถูกเก็บไปยังฐานข้อมูล แสดงดังรูปที่ 3.17



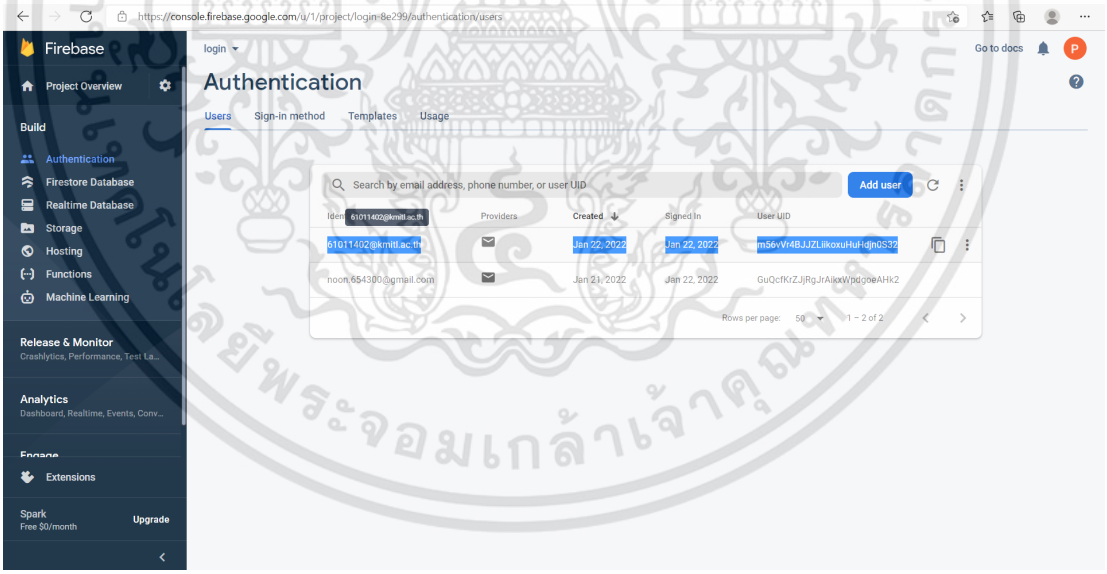
รูปที่ 3.14 ติดตั้ง Firebase บนโปรแกรมประยุกต์



รูปที่ 3.15 ฐานข้อมูล Authentication ที่สร้างขึ้น



รูปที่ 3.16 ลงทะเบียนเข้าใช้งานโปรแกรมประยุกต์



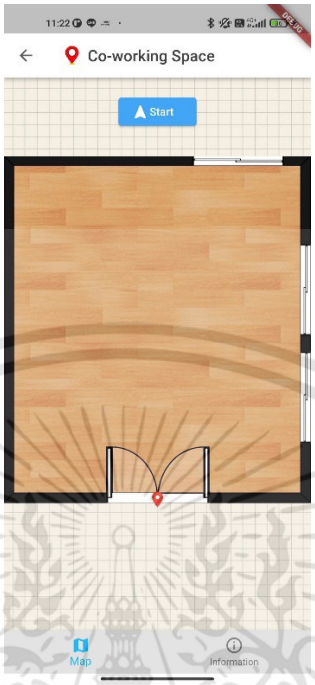
รูปที่ 3.17 ข้อมูลผู้ใช้งานโปรแกรมประยุกต์

3.1.6.2 ออกแบบโปรแกรมประยุกต์หลัก

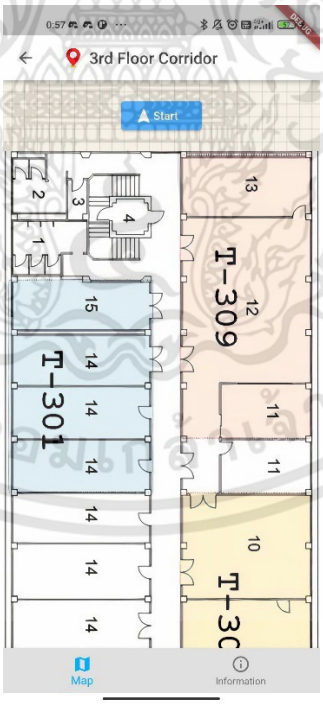
ผู้จัดทำได้ทำการออกแบบโปรแกรมประยุกต์ โดยแบ่งออกเป็น 4 หน้าหลัก ๆ คือ หน้าแรกที่จะแสดงเมื่อผู้ใช้งานทำการเข้าสู่ระบบ ซึ่งเป็นหน้าแสดงรายการสถานที่ แสดงดังรูปที่ 3.18

รูปที่ 3.18 หน้าโปรแกรมประยุกต์แสดงรายการสถานที่

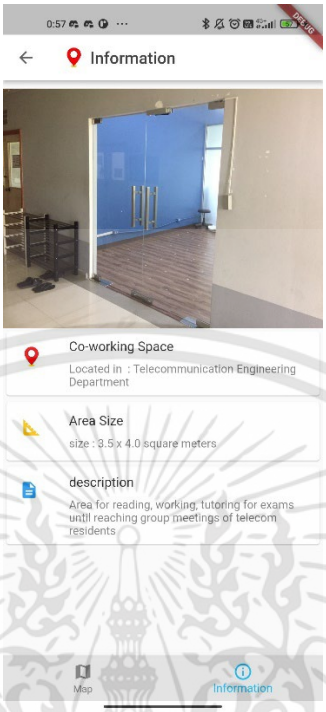
เมื่อผู้ใช้งานกดเลือกไปที่สถานที่นั้น โปรแกรมประยุกต์จะนำทางไปที่หน้าถัดไป ซึ่งเป็นหน้าแสดงแผนที่ของสถานที่ ที่ผู้ใช้งานกดเลือก แสดงดังรูปที่ 3.19 และ 3.20 โดยในหน้านี้จะมีแท็บอยู่ด้านล่าง 2 แท็บ คือ แท็บแผนที่ (Map) และแท็บข้อมูล (Information) เมื่อกดที่แท็บข้อมูลโปรแกรมประยุกต์จะนำทางไปยังหน้าข้อมูลเพิ่มเติม ที่จะแสดงข้อมูลรูปภาพสถานที่ตั้ง และคำอธิบายของสถานที่นั้น ดังรูปที่ 3.21 และ 3.22



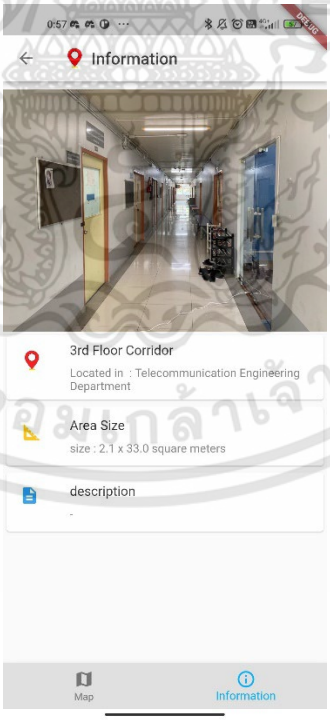
รูปที่ 3.19 หน้าโปรแกรมประยุกต์แสดงแผนที่ (Co-working Space)



รูปที่ 3.20 หน้าโปรแกรมประยุกต์แสดงแผนที่ (3rd Floor Corridor)

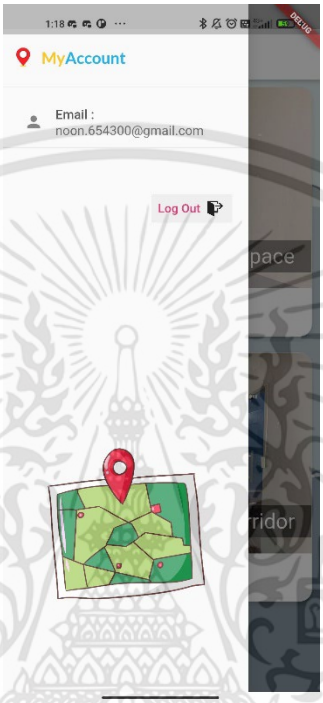


รูปที่ 3.21 หน้าโปรแกรมประยุกต์แสดงข้อมูลเพิ่มเติมของสถานที่ (Co-working Space)



รูปที่ 3.22 หน้าโปรแกรมประยุกต์แสดงข้อมูลเพิ่มเติมของสถานที่ (3rd Floor Corridor)

ในส่วนของหน้าสุดท้ายเป็นส่วนที่แสดงข้อมูลบัญชีผู้ใช้งาน โดยสามารถเข้าถึงได้จากปุ่มทางด้านบนมุมซ้ายของหน้าแรก ดังรูปที่ 3.23 ซึ่งในหน้านี้จะมีปุ่มออกจากระบบ เมื่อผู้งานกดปุ่มออกจากระบบ โปรแกรมประยุกต์ก็นำทางไปยังหน้าเริ่มต้นใช้งาน



รูปที่ 3.23 หน้าโปรแกรมประยุกต์แสดงข้อมูลบัญชีผู้ใช้

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ รุ่น ESP32-WROOM-32 เป็นไมโครคอนโทรลเลอร์ที่ใช้จะทำหน้าที่เป็น Access Point ในการส่งสัญญาณ Wi-Fi มาตรฐาน 802.11b/g/n ด้วยคลื่นความถี่ 2.4 GHz ไปยังเครื่องรับ โดยใช้พลังงานผ่านช่อง Micro USB ด้วยกระแสไฟฟ้าขั้นต่ำ 500mA ดังรูปที่ 3.23 สามารถส่งสัญญาณด้วยกำลังสูงสุด +20.5 dBm สำหรับการส่งแบบมาตรฐาน 802.11b และสูงสุด +18 dBm สำหรับการส่งแบบมาตรฐาน 802.11n ซึ่งจะทำการติดตั้งไมโครคอนโทรลเลอร์ไว้รอบ ๆ ห้องจำนวน 3 – 16 ตัว ตามจำนวนจากผลการทดสอบของระบบ

รูปที่ 3.24 ไมโครคอนโทรลเลอร์

3.3 การจัดเก็บผลการทดลอง

3.3.1 การทดสอบแบบจำลองการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

ในการที่จะเก็บข้อมูลค่า RSSI ในรูปแบบของสภาพแวดล้อมต่าง ๆ ที่หลากหลาย จำเป็นต้องใช้เวลาและการทดสอบเป็นจำนวนมาก เพื่อลดภาระเหล่านี้จึงได้สร้างแบบจำลองขึ้นมาเพื่อประเมินแนวโน้มและความเป็นไปได้ของผลลัพธ์ แล้วจึงใช้ข้อมูลที่ได้จากการทดสอบนำมาสร้างแบบจำลองการระบุตำแหน่งภายในขอบเขตและพารามิเตอร์ของการทดสอบที่ลดน้อยลง

3.3.1.1 การทดสอบผลของจำนวนเครื่องส่ง

เครื่องส่งเป็นปัจจัยสำคัญในการระบุตำแหน่ง และพารามิเตอร์หลักของแบบจำลองนี้คือจำนวนข้อมูลนำเข้าเช่นกัน หมายความว่าถ้าข้อมูลนำเข้ามีจำนวนมาก พารามิเตอร์ของผลลัพธ์ก็จะมากขึ้นตาม จำนวนเครื่องส่งที่มากขึ้นจึงมีแนวโน้มที่จะทำให้ความแม่นยำของผลลัพธ์มากขึ้นด้วย

3.3.1.2 การทดสอบผลของความละเอียดในการระบุพิกัด

ความละเอียดและความแม่นยำในการระบุตำแหน่งด้วย RSSI นั้นถูกจำกัดด้วยสัญญาณรบกวน เมื่อทำการปรับปรุงระบบด้วยวิธีการต่าง ๆ จะทำให้ความละเอียดและความแม่นยำสูงขึ้น แต่ก็ยังมีข้อจำกัดของระบบ จึงทดสอบหาจุดขีดสุดของระบบที่สามารถทำได้

3.3.1.3 การทดสอบผลของ Hyperparameter

ในการใช้งานโครงข่ายเซลล์ประสาทนั้นมีพารามิเตอร์ที่เรียกว่า Hyperparameter ที่ต้องกำหนดค่าในการฝึกฝนแบบจำลอง ซึ่งส่งผลต่อความแม่นยำของแบบจำลองแตกต่างกันตามชนิดและรูปแบบของข้อมูลที่ใส่ลงไปในโครงข่ายเซลล์ประสาท

3.3.1.4 การทดสอบแบบจำลองการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

เมื่อพบจุดขีดสุดของระบบที่เป็นไปได้จากการทดสอบพารามิเตอร์ต่าง ๆ จะถูกนำมาวิเคราะห์แล้วกำหนดพารามิเตอร์บางชนิด รวมทั้งลดขอบเขตของพารามิเตอร์ในการทดสอบที่ไร้ประสิทธิภาพลง เพื่อลดภาระของการจัดทำแบบทดสอบ จากนั้นจึงทดสอบด้วยการกำหนดและลดพารามิเตอร์เหล่านี้

3.3.1.5 การทดสอบการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาทภายในห้องทดลอง

เมื่อได้ผลลัพธ์จากการทดสอบแบบจำลอง จึงนำมาสู่การกำหนดขอบเขตของพารามิเตอร์สำหรับการระบุตำแหน่งภายในห้องทดลอง แล้วทำการทดสอบด้วยพารามิเตอร์ที่กำหนด แล้วประเมินประสิทธิภาพของระบบ เพื่อสร้างต้นแบบของระบบการระบุตำแหน่งภายในอาคาร

3.3.2 การทดสอบหาพารามิเตอร์สำหรับแบบจำลอง Path Loss

เนื่องจากการจำลองชุดข้อมูลก่อนหน้าที่จะมีการทดสอบนี้ ได้มีการนำข้อมูลการทดสอบของความสัมพันธ์ระหว่าง RSSI กับระยะทางมาจากเอกสารอ้างอิง จึงได้ทำการทดสอบด้วยตัวเอง โดยการเก็บค่า RSSI จากเครื่องส่งไมโครคอนโทรลเลอร์ ESP32 ที่ส่งมายังเครื่องรับโทรศัพท์เคลื่อนที่ด้วยโปรแกรมประยุกต์ที่ออกแบบมาสำหรับการบันทึกค่า

3.3.3 การทดสอบการปรับปรุงระบบการระบุตำแหน่ง

3.3.3.1 การใช้งานเครื่องส่งความถี่ 5 GHz

เนื่องจากมาตรฐาน WiFi มีทั้งความถี่ 2.4 GHz และ 5 GHz จึงได้ทดสอบเปรียบเทียบทั้งสองความถี่ ด้วยการบันทึกค่า RSSI จากเครื่องส่งแต่ละความถี่ในระยะทางต่าง ๆ ซึ่งผลลัพธ์ที่ได้อาจจะเป็นประโยชน์ในการปรับปรุงระบบ

3.3.3.2 การใช้งานระบบจำแนกข้อมูลอนุกรมด้วยการเรียนรู้เชิงลึก

ด้วยลักษณะของชุดข้อมูลที่บันทึกต่อเนื่องตามโดเมนเวลา สามารถนำมาแบ่งออกเป็นชุดอนุกรมได้ จึงนำมาสู่การทดสอบด้วยวิธีจำแนกข้อมูลอนุกรมด้วยการเรียนรู้เชิงลึก ซึ่งผลลัพธ์อาจจะต่างไปจากการใช้ระบบโครงข่ายเซลล์ประสาท

3.3.3.3 การประยุกต์ใช้งานภายในอาคาร

ทำการออกแบบการติดตั้งภายในอาคาร โดยกำหนดพื้นที่บริเวณทางเดินภายในอาคาร เลือกใช้จำนวนเครื่องส่งและอัตราส่วนที่เหมาะสม ฝึกฝนด้วย Hyperparameter ที่เหมาะสมจากผลการทดสอบแบบจำลอง เพื่อสร้างระบบการระบุตำแหน่งภายในอาคาร

3.3.4 ทดสอบการทำงานของเซิร์ฟเวอร์ประมวลผล

เซิร์ฟเวอร์ประมวลผลมีหน้าที่ในการประมวลผลการระบุตำแหน่งและสื่อสารกับโปรแกรมประยุกต์ จึงออกแบบเซิร์ฟเวอร์ให้มีการรับข้อมูลจากโปรแกรมประยุกต์ และทำการระบุตำแหน่งแล้วส่งข้อมูลกลับไปยังโปรแกรมประยุกต์ โดยคำสั่งในโปรแกรมส่วนแรกคือระบบการระบุตำแหน่ง ดังรูปที่ 3.25 และส่วนที่สองคือ API สำหรับการติดต่อกับโปรแกรมประยุกต์ ดังรูปที่ 3.26

```
10
11  app = FastAPI()
12
13  def findPosition (rssi):
14      path = ''
15      dataframe = pd.read_csv(path+'new_0.3_20.csv')
16      dependent_variable = 'position'
17      continuous_column_names = ['rssi_0', 'rssi_1', 'rssi_2', 'rssi_3', 'rssi_4', 'rssi_5',
18                                 'rssi_6', 'rssi_7', 'rssi_8', 'rssi_9', 'rssi_10', 'rssi_11']
19      preprocesses = [FillMissing, Normalize]
20      idx_80 = len(dataframe) - int(len(dataframe)*0.2)
21      valid_idx = list(range(idx_80, len(dataframe)))
22      batchsize = 640
23      nh = [100, 60, 60]
24      tabularlist_test = TabularList.from_df(dataframe.iloc[idx_80:len(dataframe)].copy(),
25                                             path=path,
26                                             cont_names=continuous_column_names,
27                                             procs=preprocesses)
28      databunch = (TabularList.from_df(dataframe, path=path,
29                                       cont_names=continuous_column_names,
30                                       procs=preprocesses)
31                               .split_by_idx(valid_idx)
32                               .label_from_df(cols=dependent_variable)
33                               .add_test(tabularlist_test)
34                               .databunch(bs=batchsize, num_workers=0))
35      learner = tabular_learner(databunch, layers=nh,
36                                metrics=accuracy,
37                                callback_fns=ShowGraph)
38      learner.load("trainvm")
39      return learner.predict(rssi)
40
```

รูปที่ 3.25 คำสั่งระบบการระบุตำแหน่งบนเซิร์ฟเวอร์

```python
class MyValue:
    def __init__(self, rssi, pos):
        self.rssi = rssi
        self.pos = pos

@app.get("/rssi/")
async def getrssi():
    return MyValue.rssi

@app.get("/pos/")
async def getpos():
    return MyValue.pos

@app.post("/rssi/")
async def postrssi(input: RSSI):
    MyValue.rssi = input.dict()
    rssi = input.dict()
    loc = int(findPosition(rssi)[0])
    path = ''
    map = pd.read_csv(path+'new_0.3_20_RSSI_Dataset.csv')
    x = map['x'][loc]
    y = map['y'][loc]
    MyValue.pos = {"x": int(x), "y": int(y)}
    return {"x": int(x), "y": int(y)}

if __name__ == '__main__':
    uvicorn.run(app, host='http://161.246.18.222', port=8888)
```

รูปที่ 3.26 คำสั่งของ API สำหรับการติดต่อกับโปรแกรมประยุกต์

3.3.5 การทดสอบการทำงานของโปรแกรมประยุกต์

เมื่อส่วนนำเข้าข้อมูลรับค่า RSSI จากเครื่องส่งแต่ละตัว และส่งต่อไปยังเซิร์ฟเวอร์แล้ว เซิร์ฟเวอร์จะนำข้อมูลที่ได้รับไปประมวลผลแล้วส่งที่เป็นค่าพิกัดกลับมาแสดงผลให้กับผู้ใช้งาน โดยจะทดสอบให้โปรแกรมประยุกต์ส่งชุดข้อมูลของค่า RSSI ไปยังเซิร์ฟเวอร์ จากนั้นให้เซิร์ฟเวอร์ทำการประมวลผล และส่งค่าที่เป็นพิกัด (x, y) กลับมาแสดงบนโปรแกรมประยุกต์

บทที่ 4

ผลการทดลอง

4.1 การทดสอบการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

ในการทดสอบการทำนายตำแหน่งด้วยโครงข่ายเซลล์ประสาทได้มีการใช้งานไลบรารี fastai โดยมี Preprocessing ด้วยการใช้ Fill Missing เพื่อเติมค่า RSSI ที่ขาดหายไป ซึ่งอาจจะเกิดขึ้นจากการรับสัญญาณไม่ได้หรือปัญหาอื่น ๆ ที่ไม่คาดคิด และ Normalize เพื่อให้ค่าของข้อมูลอยู่ในช่วงระดับที่ระบบสามารถนำไปใช้งานได้ ทำการแบ่งจำนวนชุดข้อมูลด้วยอัตราส่วนเป็น 80:20 สำหรับการฝึกและทดสอบแบบจำลอง จากนั้นกำหนดจำนวน Hidden Layer จำนวนเซลล์ประสาท จำนวนการฝึกฝน อัตราการเรียนรู้ และขนาด Batch Size โดยให้ข้อมูลในคอลัมน์ rssi_0, ..., rssi_n คือข้อมูลนำเข้า และข้อมูลในคอลัมน์ position คือข้อมูลส่งออก โดยกำหนดให้ข้อมูลนำเข้าเป็นชนิดค่าต่อเนื่อง (Continuous) ทั้งหมด เพราะค่า RSSI นั้นเปลี่ยนแปลงต่อเนื่องตามระยะทาง ประสิทธิภาพและความแม่นยำของแบบจำลองนั้นขึ้นอยู่ตัวแปรหลายอย่างที่จะทดสอบดังต่อไปนี้

4.1.1 ผลของจำนวนเครื่องส่ง

จำนวนเครื่องส่งที่มากขึ้นส่งผลให้ข้อมูลนำเข้าที่จะป้อนเข้าสู่ โครงข่ายเซลล์ประสาทก็มากขึ้นด้วย ข้อมูลนำเข้าที่มากขึ้นทำให้การทำนายข้อมูลส่งออกนั้นมีความละเอียดแม่นยำมากขึ้นด้วย จึงได้ทำการทดสอบด้วยการจำลองเครื่องส่งตั้งแต่ 3 – 16 ตัว โดยใช้ชุดข้อมูลของห้องขนาด 4.8 x 4.8 ตารางเมตร ที่มีอัตราส่วนระยะห่างต่อพิกัดเท่ากับ 0.40 เมตร จำนวน 169 ตำแหน่ง มีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 100 180 และ 220 ตามลำดับ มีขนาด Batch Size เท่ากับ 32 ใช้อัตราการเรียนรู้เท่ากับ 0.0003 และมีจำนวนการฝึกฝนเท่ากับ 500 รอบ แล้วทำการสร้างแบบจำลองในแต่ละแบบเพื่อที่จะประเมินผลลัพธ์ที่ได้ จากผลลัพธ์ที่ได้ดังตารางที่ 4.1 นั้นสรุปได้ว่า จำนวนเครื่องส่งที่มากขึ้นนั้นส่งผลทำให้การระบุตำแหน่งมีความแม่นยำขึ้น แต่ถึงอย่างนั้นจำนวนเครื่องส่งที่มากขึ้นก็ส่งผลให้มีการติดตั้งที่ยุ่งยากขึ้นตามมาด้วย อีกทั้งค่าใช้จ่ายที่สูงขึ้นจึงต้องพิจารณาถึงความเหมาะสมและความคุ้มค่าต่อการใช้งาน

ตารางที่ 4.1 ผลของจำนวนเครื่องส่งในการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

จำนวนเครื่องส่ง (ตัว)	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	45	0.6878	0.0000	7.6158	0.7680
4	61	0.4615	0.0000	3.6056	0.6020
5	69	0.3513	0.0000	2.8284	0.5377
6	73	0.2976	0.0000	12.6491	0.5786
7	80	0.2125	0.0000	2.8284	0.4330
8	83	0.1819	0.0000	2.8284	0.4173
9	86	0.1573	0.0000	12.0000	0.4776
10	88	0.1408	0.0000	8.5440	0.5210
11	89	0.1362	0.0000	7.2111	0.4913
12	92	0.0851	0.0000	1.4142	0.2907
13	92	0.0746	0.0000	1.4142	0.2661
14	96	0.0448	0.0000	1.4142	0.2102
15	95	0.0663	0.0000	11.4018	0.4076
16	96	0.0403	0.0000	1.4142	0.2010

4.1.2 ผลของความละเอียดในการระบุพิกัด

1) ผลของขนาดพื้นที่

ขนาดของพื้นที่ที่กว้างขึ้นย่อมมีจำนวนพิกัดมากขึ้นหากใช้อัตราส่วนคงที่ จึงต้องมีการทดสอบหาช่วงขนาดพื้นที่ที่ส่งผลให้มีความแม่นยำเหมาะสมเพียงพอต่อการนำไปใช้งานจริง ในการทดสอบได้จำลองขนาดห้องตั้งแต่ 3.6 x 3.6 ถึง 12.0 x 12.0 ตารางเมตร หรือมีจำนวนพิกัดตั้งแต่ 100 – 961 ตำแหน่ง กำหนดให้มีอัตราส่วนระยะห่างต่อพิกัดเท่ากับ 0.40 เมตร ใช้เครื่องส่ง 8 ตัว มีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 100 180 และ 220 ตามลำดับ มีขนาด Batch Size เท่ากับ 32 ใช้อัตราการเรียนรู้เท่ากับ 0.0003 และมีจำนวนการฝึกฝนเท่ากับ 500 รอบ แล้วทำการสร้างแบบจำลองในแต่ละแบบเพื่อที่จะประเมินผลลัพธ์ที่ได้ จากผลลัพธ์ที่ได้

ดังตารางที่ 4.2 นั้นสรุปได้ว่า พื้นที่ที่กว้างขึ้นส่งผลให้ความแม่นยำลดลงตามมา เพื่อแก้ปัญหาเมื่อใช้งานกับห้องที่มีขนาดใหญ่จึงต้องเพิ่มเครื่องส่งให้มากขึ้นด้วย

ตารางที่ 4.2 ผลของขนาดพื้นที่ในการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

พื้นที่ (ตารางเมตร)	จำนวน ตำแหน่ง	ความ แม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
			เฉลี่ย	น้อยสุด	มากสุด	ส่วน เบี่ยงเบน มาตรฐาน
3.6 x 3.6	100	94	0.0591	0.0000	1.4142	0.2481
4.8 x 4.8	169	83	0.1819	0.0000	2.8284	0.4173
6.0 x 6.0	256	66	0.3747	0.0000	6.4031	0.5526
7.2 x 7.2	361	53	0.5292	0.0000	7.0711	0.5911
8.4 x 8.4	484	41	0.7121	0.0000	24.6982	0.9201
9.6 x 9.6	625	33	0.8715	0.0000	25.9422	1.2540
10.8 x 10.8	784	22	1.0477	0.0000	21.8403	0.8560
12.0 x 12.0	961	22	1.0896	0.0000	32.2490	1.2557

2) ผลของอัตราส่วนต่อขนาดพื้นที่

อัตราส่วนต่อขนาดพื้นที่คือระยะห่างระหว่างจุดพิกัดสองจุดเพื่อใช้เป็นตัวบอกระยะจริง ในระบบของการระบุตำแหน่งนั้นจะใช้พิกัดที่สร้างขึ้นแทนตำแหน่งนั้น ๆ ซึ่งเป็นตัวแปรสำคัญในการระบุตำแหน่ง โดยถ้าหากใช้อัตราส่วนน้อยหรือกล่าวคือระยะห่างระหว่างพิกัดแคบ จะทำให้มีความละเอียดในการระบุตำแหน่งสูง แต่ถ้าหากใช้อัตราส่วนมากหรือระยะห่างระหว่างพิกัดกว้างจะทำให้มีความละเอียดในการระบุตำแหน่งต่ำ ซึ่งในทางปฏิบัติเป็นไปได้ยากที่ค่า RSSI จะมีการเปลี่ยนแปลงในระยะทางที่น้อยมาก ๆ จึงมีความท้าทายที่จะระบุตำแหน่งในความละเอียดสูง โดยการทดสอบจะเปรียบเทียบอัตราส่วนตั้งแต่ 0.2 – 0.9 ให้มีขนาดพื้นที่ 100±2.9 ตารางเมตร ใช้เครื่องส่ง 8 ตัว มีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 100 180 และ 220 ตามลำดับ มีขนาด Batch Size เท่ากับ 32 ใช้อัตราการฝึกฝนเท่ากับ 0.0003 และมีจำนวนการฝึกฝนเท่ากับ 500 รอบ แล้วทำการสร้างแบบจำลองในแต่ละแบบเพื่อที่จะประเมินผลลัพธ์ที่ได้จากผลลัพธ์ที่ได้ดังตารางที่ 4.3 นั้นสรุปได้ว่า อัตราส่วนที่มากขึ้นจะทำให้ความแม่นยำของ

แบบจำลองสูงขึ้นด้วย แต่ในความแม่นยำที่สูงขึ้นนี้มีผลเสียคืออัตราส่วนที่มากขึ้น ทำให้ความละเอียดในการระบุตำแหน่งต่ำลง จึงต้องพิจารณาถึงเป้าหมายและความเป็นไปได้ที่จะใช้ความละเอียดนั้น ๆ ก่อนที่จะนำไปปรับปรุงใช้งาน

ตารางที่ 4.3 ผลของอัตราส่วนต่อขนาดพื้นที่ในการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

อัตราส่วน	จำนวนตำแหน่ง	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
			เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
0.2	2601	13	1.6753	0.0000	55.9732	2.1319
0.3	1190	25	1.0891	0.0000	33.2415	1.1866
0.4	676	39	0.7650	0.0000	17.6918	0.7982
0.5	441	56	0.5093	0.0000	13.4164	0.6676
0.6	306	68	0.3514	0.0000	2.8284	0.5324
0.7	240	78	0.2402	0.0000	3.6056	0.4627
0.8	182	86	0.1503	0.0000	4.1231	0.3852
0.9	144	93	0.0768	0.0000	1.4142	0.2833

4.1.3 ผลของ Hyperparameter

1) ผลของ Batch Size

Batch Size คือ ขนาดของกลุ่มข้อมูลขนาดเล็กที่จะใส่เข้าไปเพื่อฝึกแบบจำลอง เนื่องจากไม่สามารถใส่ข้อมูลเข้าไปประมวลผลทั้งหมดทีเดียวได้ Batch Size ที่ใหญ่กว่าจะใช้หน่วยความจำที่มากกว่า และทำให้ประมวลผลเร็วกว่า อีกทั้งยังส่งผลต่อความแม่นยำของแบบจำลองอีกด้วย จึงได้ทดสอบขนาด Batch Size ตั้งแต่ 4 – 128 โดยใช้ชุดข้อมูลของห้องขนาด 4.8 x 4.8 ตารางเมตร ที่มีอัตราส่วนเท่ากับ 0.40 จำนวน 169 ตำแหน่ง มีจำนวนเครื่องส่ง 8 ตัว มีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 100 180 และ 220 ตามลำดับ ใช้อัตราการฝึกฝน เท่ากับ 0.0003 และมีจำนวนการฝึกฝนเท่ากับ 500 รอบ แล้วทำการสร้างแบบบจำรองในแต่ละแบบเพื่อที่จะประเมินผลลัพธ์ที่ได้ จากผลลัพธ์ที่ได้ดังตารางที่ 4.4 นั้นสรุปได้ว่าขนาด Batch Size ส่งผลต่อความแม่นยำ โดยความแม่นยำมีแนวโน้มเพิ่มขึ้นตามขนาด Batch Size จนถึง

จุด ๆ หนึ่งแล้วจึงลดลง เพราะฉะนั้น ในการฝึกฝนแบบจำลองในแบบจำลองอื่น ๆ ควรจะทำการทดลองซ้ำเสมอ เพื่อหาขนาด Batch Size ที่ดีที่สุด

ตารางที่ 4.4 ผลของ Batch Size ในการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

Batch Size	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
4	31	1.052	0.000	11.402	1.472
8	26	1.077	0.000	12.649	1.445
16	69	0.331	0.000	2.828	0.506
32	83	0.182	0.000	2.828	0.417
64	88	0.128	0.000	2.828	0.359
128	88	0.136	0.000	2.828	0.370

2) ผลของจำนวนการฝึกฝน

จำนวนการฝึกฝน คือ จำนวนรอบที่ใส่ข้อมูลทั้งชุดเข้าไปจนครบของการฝึกฝนแบบจำลอง โดยแต่ละรอบจะทำให้ Loss ลดลง ในขณะที่ความแม่นยำเพิ่มขึ้น ยิ่งใช้จำนวนรอบมากเวลาในการประมวลก็จะนานขึ้นด้วย ด้วยเหตุนี้จึงต้องใช้จำนวนการฝึกฝนที่เหมาะสมเพื่อประหยัดเวลาในการประมวลผล จึงได้ทำการทดสอบจำนวนการฝึกฝนตั้งแต่ 100 – 1000 รอบ โดยใช้ชุดข้อมูลของห้องขนาด 4.8 x 4.8 ตารางเมตร ที่มีอัตราส่วน 0.40 จำนวน 169 ตำแหน่ง มีจำนวนเครื่องส่ง 8 ตัว มีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 100 180 และ 220 ตามลำดับ มีขนาด Batch Size เท่ากับ 32 และใช้อัตราการเรียนรู้เท่ากับ 0.0003 แล้วทำการสร้างแบบจำลองในแต่ละแบบเพื่อที่จะประเมินผลลัพธ์ที่ได้ จากผลลัพธ์ที่ได้ดังตารางที่ 4.5 นั้นสรุปได้ว่า จำนวนการฝึกฝนที่มากขึ้นทำให้แบบจำลองมีความแม่นยำเพิ่มขึ้น จนกระทั่งถึงจุดที่ทำให้ความแม่นยำลดลง ซึ่งเกิดจาก overfitting นอกจากนี้จำนวนการฝึกฝนยังสัมพันธ์กับอัตราการเรียนรู้เนื่องจากอัตราการเรียนรู้ที่สูงจะทำให้ Loss เกิดการเปลี่ยนแปลงอย่างรวดเร็ว ทำให้ความแม่นยำเพิ่มขึ้นอย่างรวดเร็วโดยที่ใช้จำนวนรอบการฝึกฝนที่น้อย ในขณะที่ใช้อัตราการเรียนรู้ต่ำจะทำให้ Loss เกิดการเปลี่ยนแปลงอย่างช้า ๆ ทำให้ความแม่นยำเพิ่มขึ้นช้าตามไปด้วย จึงต้องใช้จำนวนรอบการฝึกฝนที่มากขึ้นเพื่อให้ค่าความแม่นยำลู่เข้าสู่จุดที่เหมาะสม แต่ถ้าหากใช้จำนวนรอบการฝึกฝนที่มากเกินไปก็อาจจะส่งผลให้เกิด Overfitting ได้ซึ่งไม่เป็นผลดี

ตารางที่ 4.5 ผลของจำนวนการฝึกฝนในการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

จำนวนการฝึกฝน (รอบ)	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
100	82	0.1896	0.0000	1.4142	0.4153
200	85	0.1682	0.0000	4.0000	0.4080
300	83	0.1809	0.0000	2.8284	0.4098
400	84	0.1848	0.0000	11.3137	0.4983
500	84	0.1740	0.0000	2.8284	0.4078
600	84	0.1728	0.0000	2.8284	0.4047
700	82	0.1919	0.0000	2.8284	0.4206
800	84	0.1718	0.0000	2.8284	0.4021
900	79	0.2345	0.0000	2.8284	0.4598
1000	79	0.2250	0.0000	2.8284	0.4522

3) ผลของอัตราการเรียนรู้

อัตราการเรียนรู้ คือ อัตราการเรียนรู้ที่จะเป็นตัวควบคุมการเปลี่ยนแปลงของการฝึกแบบจำลอง ถ้าอัตราการเรียนรู้สูง น้ำหนักของแบบจำลองก็จะเปลี่ยนแปลงมาก ทำให้ Loss เปลี่ยนแปลงมาก ในทางกลับกันถ้าหากอัตราการเรียนรู้ต่ำ น้ำหนักของแบบจำลองก็จะเปลี่ยนแปลงน้อย ทำให้ Loss เปลี่ยนแปลงน้อยเช่นกัน นอกจากนี้ยังส่งผลต่อความแม่นยำอีกด้วยถึงแม้จะกำหนดอัตราการเรียนรู้และจำนวนการฝึกฝนให้สัมพันธ์กันก็ตาม จึงได้ทดสอบหาความแตกต่างของค่าอัตราการเรียนรู้ตั้งแต่ 0.00001 – 0.01 ในการทดสอบได้จำลองชุดข้อมูลของห้องขนาด 4.8 x 4.8 ตารางเมตร ที่มีอัตราส่วน 0.40 จำนวน 169 ตำแหน่ง มีจำนวนเครื่องส่ง 8 ตัว มีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 100 180 และ 220 ตามลำดับ มีขนาด Batch Size เท่ากับ 32 และมีจำนวนการฝึกฝนเท่ากับ 500 รอบ แล้วทำการสร้างแบบจำลองในแต่ละแบบเพื่อที่จะประเมินผลลัพธ์ที่ได้ จากผลลัพธ์ที่ได้ดังตารางที่ 4.6 นั้นสรุปได้ว่าอัตราการเรียนรู้ที่มากขึ้นไม่เป็นผลดีต่อแบบจำลอง เพราะทำให้ความแม่นยำลดลงเป็นอย่างมาก แต่อัตราการเรียนรู้ที่ต่ำเกินไปก็ไม่เป็นผลดี เพราะใช้เวลาในการฝึกฝนแบบจำลองที่นานมาก อีกทั้งยังต้องใช้

จำนวนรอบการฝึกฝนที่มากอีกด้วย โดยอัตราการเรียนรู้ที่ 0.00005 – 0.0005 นั้นให้ผลลัพธ์ที่ดี ใช้จำนวนรอบการฝึกฝนที่ไม่มาก และยังมีความเร็วในการประมวลผลที่เหมาะสมอีกด้วย

ตารางที่ 4.6 ผลของอัตราการเรียนรู้ในการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

อัตราการเรียนรู้	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
0.010000	75	0.3611	0.0000	12.6491	1.0664
0.007500	73	0.2927	0.0000	7.0711	0.5151
0.005000	76	0.2605	0.0000	2.8284	0.4739
0.002500	74	0.2780	0.0000	2.8284	0.4857
0.001000	79	0.2240	0.0000	2.8284	0.4468
0.000750	84	0.2335	0.0000	2.2361	0.4512
0.000500	83	0.1994	0.0000	2.8284	0.4297
0.000250	85	0.1617	0.0000	2.8284	0.3945
0.000100	79	0.1583	0.0000	1.4142	0.3837
0.000075	85	0.1677	0.0000	2.8284	0.4009
0.000050	85	0.1652	0.0000	1.4142	0.3915
0.000025	81	0.2075	0.0000	8.5440	0.4751
0.000010	81	0.2005	0.0000	2.2361	0.4264

4) ผลของจำนวน Hidden Layer และจำนวนเซลล์ประสาท

โดยพื้นฐานถ้าหากต้องการความแม่นยำที่มากขึ้น ก็จะต้องเพิ่มจำนวน Hidden Layer และจำนวนเซลล์ประสาทให้มากขึ้น แต่ก็ไม่เสมอไปเพราะข้อมูลนั้นมีหลากหลายชนิด หากใช้จำนวน Hidden Layer และจำนวนเซลล์ประสาทที่มากเกินไปก็อาจจะทำให้เกิด Overfitting จึงได้ทดสอบจำนวน Hidden Layer และจำนวนเซลล์ประสาทที่แตกต่างกัน เพื่อหาผลลัพธ์ที่ดี ในการทดสอบได้จำลองชุดข้อมูลของห้องขนาด 4.8 x 4.8 ตารางเมตร ที่มีอัตราส่วน 0.40 จำนวน 169 ตำแหน่ง มีจำนวนเครื่องส่ง 8 ตัว มีขนาด Batch Size เท่ากับ 64 ใช้อัตราการเรียนรู้เท่ากับ 0.0003 และมีจำนวนการฝึกฝนเท่ากับ 500 รอบ แล้วทำการสร้างแบบจำลองในแต่ละแบบเพื่อที่จะประเมินผลลัพธ์ที่ได้ จากผลลัพธ์ที่ได้ดังตารางที่ 4.7 นั้นสรุปได้ว่า จำนวน Hidden

Layer ที่มากขึ้นส่งผลให้ความแม่นยำเพิ่มขึ้น และจำนวนเซลล์ประสาทที่มากขึ้นส่งผลให้ความแม่นยำเพิ่มขึ้นจนถึงจุด ๆ หนึ่ง แล้วจึงลดลง ทั้งนี้จำนวน Hidden Layer และจำนวนเซลล์ประสาทที่มากเกินไปก็จะทำให้เกิด Overfitting

ตารางที่ 4.7 ผลของจำนวน Hidden Layer และจำนวนเซลล์ประสาทในการระบุตำแหน่ง

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
1	20	-	-	78	0.2302	0.0000	2.2361	0.4436
1	40	-	-	83	0.1813	0.0000	2.2361	0.4097
1	60	-	-	86	0.1555	0.0000	1.4142	0.3825
1	80	-	-	84	0.1724	0.0000	1.4142	0.3967
1	100	-	-	82	0.1885	0.0000	1.4142	0.4136
1	120	-	-	84	0.1695	0.0000	2.2361	0.3972
1	140	-	-	85	0.1668	0.0000	1.4142	0.3968
1	160	-	-	84	0.1695	0.0000	1.4142	0.3964
1	180	-	-	82	0.1912	0.0000	2.2361	0.4124
1	200	-	-	84	0.1669	0.0000	1.4142	0.3938
1	220	-	-	86	0.1530	0.0000	1.4142	0.3789
2	20	20	-	87	0.1480	0.0000	12.3693	0.4718
2	20	40	-	85	0.1572	0.0000	1.4142	0.3849
2	20	60	-	86	0.1491	0.0000	1.4142	0.3734
2	20	80	-	85	0.1582	0.0000	1.4142	0.3845
2	20	100	-	85	0.1631	0.0000	2.2361	0.3909
2	20	120	-	85	0.1578	0.0000	1.4142	0.3823
2	40	20	-	86	0.1562	0.0000	2.2361	0.3853
2	40	40	-	87	0.1402	0.0000	1.4142	0.3648
2	40	60	-	87	0.1388	0.0000	2.8284	0.3662
2	40	80	-	85	0.1642	0.0000	1.4142	0.3919

ตารางที่ 4.7 ผลของจำนวน Hidden Layer และจำนวนเซลล์ประสาทในการระบุตำแหน่ง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
2	40	100	-	87	0.1416	0.0000	2.0000	0.3683
2	40	120	-	85	0.1563	0.0000	1.4142	0.3837
2	60	20	-	86	0.1623	0.0000	12.0000	0.4808
2	60	40	-	87	0.1396	0.0000	2.8284	0.3699
2	60	60	-	87	0.1385	0.0000	2.8284	0.3687
2	60	80	-	87	0.1439	0.0000	1.4142	0.3698
2	60	100	-	88	0.1322	0.0000	1.4142	0.3563
2	60	120	-	88	0.1372	0.0000	1.4142	0.3619
2	80	20	-	86	0.1518	0.0000	1.4142	0.3778
2	80	40	-	86	0.1529	0.0000	1.4142	0.3774
2	80	60	-	87	0.1360	0.0000	2.2361	0.3648
2	80	80	-	87	0.1409	0.0000	2.0000	0.3686
2	80	100	-	87	0.1457	0.0000	1.4142	0.3739
2	80	120	-	88	0.1298	0.0000	2.8284	0.3597
2	100	20	-	87	0.1428	0.0000	2.2361	0.3742
2	100	40	-	86	0.1541	0.0000	2.0000	0.3792
2	100	60	-	87	0.1379	0.0000	1.4142	0.3616
2	100	80	-	87	0.1374	0.0000	1.4142	0.3634
2	100	100	-	87	0.1391	0.0000	1.4142	0.3652
2	100	120	-	88	0.1259	0.0000	2.2361	0.3520
2	120	20	-	87	0.1431	0.0000	7.6158	0.4125
2	120	40	-	87	0.1413	0.0000	5.0000	0.3826
2	120	60	-	88	0.1314	0.0000	1.4142	0.3583
2	120	80	-	88	0.1255	0.0000	1.4142	0.3487

ตารางที่ 4.7 ผลของจำนวน Hidden Layer และจำนวนเซลล์ประสาท ในการระบุตำแหน่ง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
2	120	100	-	87	0.1372	0.0000	1.4142	0.3652
2	120	120	-	88	0.1301	0.0000	1.4142	0.3546
3	20	20	20	86	0.1498	0.0000	2.2361	0.3786
3	20	20	40	85	0.1650	0.0000	2.0000	0.3924
3	20	20	60	86	0.1538	0.0000	1.4142	0.3778
3	20	20	80	87	0.1409	0.0000	2.0000	0.3686
3	20	40	20	83	0.1853	0.0000	1.4142	0.4078
3	20	40	40	86	0.1517	0.0000	1.4142	0.3786
3	20	40	60	87	0.1393	0.0000	1.4142	0.3668
3	20	40	80	88	0.1330	0.0000	1.4142	0.3585
3	20	60	20	88	0.1338	0.0000	1.4142	0.3599
3	20	60	40	86	0.1460	0.0000	1.4142	0.3714
3	20	60	60	88	0.1325	0.0000	1.4142	0.3571
3	20	60	80	88	0.1268	0.0000	1.4142	0.3491
3	20	80	20	86	0.1497	0.0000	2.8284	0.3825
3	20	80	40	87	0.1372	0.0000	2.2361	0.3676
3	20	80	60	88	0.1403	0.0000	8.9443	0.4337
3	20	80	80	88	0.1306	0.0000	2.8284	0.3569
3	40	20	20	87	0.1402	0.0000	1.4142	0.3672
3	40	20	40	86	0.1464	0.0000	1.4142	0.3736
3	40	20	60	86	0.1446	0.0000	1.4142	0.3696
3	40	20	80	87	0.1446	0.0000	1.4142	0.3719
3	40	40	20	87	0.1413	0.0000	1.4142	0.3692
3	40	40	40	87	0.1418	0.0000	2.8284	0.3738

ตารางที่ 4.7 ผลของจำนวน Hidden Layer และจำนวนเซลล์ประสาทในการระบุตำแหน่ง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	40	40	60	87	0.1445	0.0000	1.4142	0.3704
3	40	40	80	88	0.1308	0.0000	1.4142	0.3544
3	40	60	20	88	0.1330	0.0000	1.4142	0.3585
3	40	60	40	88	0.1304	0.0000	2.0000	0.3595
3	40	60	60	89	0.1147	0.0000	1.4142	0.3378
3	40	60	80	88	0.1314	0.0000	2.8284	0.3616
3	40	80	20	89	0.1231	0.0000	1.4142	0.3462
3	40	80	40	88	0.1279	0.0000	2.8284	0.3571
3	40	80	60	88	0.1247	0.0000	1.4142	0.3473
3	40	80	80	88	0.1338	0.0000	2.8284	0.3631
3	60	20	20	86	0.1594	0.0000	13.4536	0.4993
3	60	20	40	88	0.1325	0.0000	1.4142	0.3595
3	60	20	60	87	0.1391	0.0000	1.4142	0.3652
3	60	20	80	86	0.1534	0.0000	1.4142	0.3779
3	60	40	20	86	0.1485	0.0000	2.8284	0.3760
3	60	40	40	86	0.1491	0.0000	2.8284	0.3812
3	60	40	60	89	0.1296	0.0000	9.0554	0.4134
3	60	40	80	87	0.1372	0.0000	2.8284	0.3651
3	60	60	20	87	0.1435	0.0000	1.4142	0.3700
3	60	60	40	88	0.1342	0.0000	2.8284	0.3630
3	60	60	60	91	0.1020	0.0000	1.4142	0.3195
3	60	60	80	87	0.1393	0.0000	2.8284	0.3724
3	60	80	20	89	0.1366	0.0000	9.0000	0.4702
3	60	80	40	88	0.1243	0.0000	1.4142	0.3474

ตารางที่ 4.7 ผลของจำนวน Hidden Layer และจำนวนเซลล์ประสาท ในการระบุตำแหน่ง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	60	80	60	88	0.1266	0.0000	1.4142	0.3534
3	60	80	80	89	0.1175	0.0000	1.4142	0.3404
3	80	20	20	87	0.1375	0.0000	1.4142	0.3626
3	80	20	40	87	0.1392	0.0000	2.8284	0.3708
3	80	20	60	87	0.1389	0.0000	2.8284	0.3669
3	80	20	80	87	0.1407	0.0000	2.2361	0.3671
3	80	40	20	88	0.1309	0.0000	1.4142	0.3535
3	80	40	40	88	0.1301	0.0000	2.8284	0.3612
3	80	40	60	87	0.1385	0.0000	1.4142	0.3655
3	80	40	80	88	0.1272	0.0000	2.8284	0.3606
3	80	60	20	88	0.1332	0.0000	5.0000	0.3754
3	80	60	40	88	0.1363	0.0000	2.2361	0.3679
3	80	60	60	89	0.1251	0.0000	2.8284	0.3556
3	80	60	80	88	0.1245	0.0000	1.4142	0.3491
3	80	80	20	87	0.1367	0.0000	2.8284	0.3670
3	80	80	40	89	0.1153	0.0000	1.4142	0.3359
3	80	80	60	90	0.1131	0.0000	1.4142	0.3349
3	80	80	80	88	0.1253	0.0000	1.4142	0.3505

4.1.4 แบบจำลองการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาท

จากการทดสอบในข้อ 4.1.3 นั้นใช้เพียงชุดข้อมูลของอัตราส่วน 0.4 ในพื้นที่ 4.8 x 4.8 ตารางเมตร ก็สามารถทำให้ความแม่นยำไปถึง 91% ด้วยระยะคลาดเคลื่อนเฉลี่ย 0.102 เมตร ฉะนั้นจึงเพิ่มความท้าทายของการระบุตำแหน่งด้วยอัตราส่วนที่ต่ำในพื้นที่ที่ขนาดใหญ่ขึ้นด้วยแบบจำลองชุดข้อมูลของเครื่องรับจำนวน 12 ตัว ซึ่งจำนวนนี้เพียงพอและเหมาะสมต่อการนำมาใช้

ในพื้นที่ 6 x 6 ตารางเมตร ที่มีอัตราส่วนเท่ากับ 0.3 ทั้งหมด 441 ตำแหน่ง เพราะอัตราส่วนนี้มีความเป็นไปได้ที่จะให้ผลลัพธ์ที่ดี หากอัตราส่วนต่ำกว่านี้จะให้ผลลัพธ์ที่แย่เกินกว่าจะนำมาใช้งาน ถ้าหากสูงกว่านี้ก็เป็นการด้อยคุณภาพของงานลง ซึ่งต้องการที่จะระบุตำแหน่งในความละเอียดสูง แบบจำลองนี้จะถูกนำมาทดสอบหาค่า Hyperparameter ที่เหมาะสมต่อไป

1) Batch Size

ในการทดสอบนี้จะใช้ค่า Hyperparameter ที่ได้จากผลการทดสอบในข้อ 4.1.3 โดยเลือกใช้ค่าที่เหมาะสมและให้ผลลัพธ์ที่ดีในการทดสอบก่อนหน้านี้ ซึ่งก็คือจำนวน Hidden Layer เท่ากับ 3 จำนวนเซลล์ประสาทเท่ากับ 60 60 และ 60 ตามลำดับ มีจำนวนการฝึกฝนที่ 800 รอบ และใช้อัตราการเรียนรู้เท่ากับ 0.0001 แล้วทำการทดสอบ Batch Size ที่ขนาด 32 64 128 256 384 512 640 768 896 และ 1024 ให้ผลลัพธ์ดังตารางที่ 4.8 โดยขนาด Batch Size ที่เหมาะสมที่จะนำมาใช้งานกับแบบจำลองนี้คือ 640

ตารางที่ 4.8 ผลของ Batch Size ต่อแบบจำลอง

Batch Size	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
32	64.71	0.4324	0.0000	20.0000	1.1481
64	71.24	0.3087	0.0000	2.8284	0.4949
128	71.61	0.3047	0.0000	2.8284	0.4928
256	75.21	0.2681	0.0000	2.0000	0.4744
384	76.23	0.2567	0.0000	2.8284	0.4677
512	77.18	0.2470	0.0000	2.8284	0.4624
640	77.46	0.2451	0.0000	2.2361	0.4617
768	76.62	0.2521	0.0000	2.0000	0.4634
896	77.12	0.2481	0.0000	2.2361	0.4633
1024	76.55	0.2538	0.0000	2.2361	0.4659

2) จำนวนการฝึกฝน

ในการทดสอบนี้จะเป็นการทดสอบที่ต่อจากข้อ 1) โดยใช้ค่า Hyperparameter เช่นเดียวกัน แต่จะใช้ขนาด Batch Size เท่ากับ 640 ที่ได้จากการทดสอบมาก่อนหน้านี้ ส่วนจำนวนการฝึกฝนจะทดสอบที่ 100 – 2000 รอบ ให้ผลลัพธ์ดังตารางที่ 4.9 จำนวนการฝึกฝนที่เหมาะสมและให้ผลของความแม่นยำที่ดีที่สุดคือ 900 รอบ

ตารางที่ 4.9 ผลของจำนวนการฝึกฝนต่อแบบจำลอง

จำนวนการฝึกฝน (รอบ)	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
100	43.27	0.6633	0.0000	3.6056	0.6224
200	65.19	0.3807	0.0000	2.8284	0.5333
300	71.90	0.3052	0.0000	2.2361	0.4973
400	74.74	0.2745	0.0000	2.2361	0.4804
500	75.46	0.2657	0.0000	2.2361	0.4740
600	76.28	0.2578	0.0000	2.2361	0.4703
700	76.60	0.2536	0.0000	2.2361	0.4664
800	76.98	0.2502	0.0000	2.2361	0.4649
900	77.17	0.2474	0.0000	2.2361	0.4622
1000	77.03	0.2488	0.0000	2.2361	0.4627
1100	76.98	0.2500	0.0000	2.2361	0.4648
1200	77.12	0.2479	0.0000	2.2361	0.4622
1300	76.89	0.2506	0.0000	2.2361	0.4644
1400	76.76	0.2516	0.0000	2.2361	0.4644
1500	76.89	0.2506	0.0000	2.2361	0.4644
1600	76.89	0.2504	0.0000	2.2361	0.4641
1700	76.24	0.2580	0.0000	2.2361	0.4696
1800	76.12	0.2583	0.0000	2.2361	0.4685
1900	76.33	0.2564	0.0000	2.2361	0.4676
2000	76.15	0.2592	0.0000	2.2361	0.4707

3) อัตราการเรียนรู้

ในการทดสอบนี้จะเป็นการทดสอบที่ต่อจากข้อ 2) โดยใช้ค่า Hyperparameter เช่นเดียวกัน ซึ่งก็คือมีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 60 60 และ 60 ตามลำดับ และมีขนาด Batch Size เท่ากับ 640 แต่จะใช้จำนวนการฝึกฝนที่ 900 รอบ ที่ได้มาจากการทดสอบก่อนหน้านี้ ส่วนอัตราการเรียนรู้จะทดสอบที่ 0.0005 0.0004 0.0003 0.0002 0.0001 และ 0.00005 ให้ผลลัพธ์ดังตารางที่ 4.10 อัตราการเรียนรู้ที่ให้ผลลัพธ์ดีที่สุดคือ 0.0001

ตารางที่ 4.10 ผลของอัตราการเรียนรู้ต่อแบบจำลอง

อัตราการเรียนรู้	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
0.0005	75.56	0.2643	0.0000	2.0000	0.4721
0.0004	76.62	0.2528	0.0000	2.2361	0.4652
0.0003	77.03	0.2473	0.0000	2.8284	0.4603
0.0002	77.19	0.2468	0.0000	2.8284	0.4616
0.0001	77.44	0.2438	0.0000	2.0000	0.4587
0.00005	74.67	0.2756	0.0000	2.2361	0.4814

4) Hidden Layer

ในการทดสอบนี้จะเป็นการทดสอบที่ต่อจากข้อ 3) โดยใช้ค่า Hyperparameter เช่นเดียวกัน ซึ่งก็คือใช้ขนาด Batch Size เท่ากับ 640 มีจำนวนการฝึกฝนที่ 900 รอบ แต่จะใช้อัตราการเรียนรู้เท่ากับ 0.0001 ที่ได้มาจากการทดสอบก่อนหน้านี้ ส่วน Hidden Layer ได้ทำการทดสอบจำนวนเซลล์ประสาทตั้งแต่ 60 – 120 ให้ผลลัพธ์ดังตารางที่ 4.11 ที่เซลล์ประสาทเท่ากับ 100 60 และ 60 ตามลำดับนั้นให้ความแม่นยำมากที่สุด

ตารางที่ 4.11 ผลของ Hidden Layer ต่อแบบจำลอง

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	60	60	60	76.49	0.2545	0.0000	2.8284	0.4670
3	60	60	80	78.05	0.2389	0.0000	2.8284	0.4586
3	60	60	100	77.28	0.2470	0.0000	2.8284	0.4632
3	60	60	120	76.87	0.2513	0.0000	2.8284	0.4665
3	60	80	60	78.21	0.2354	0.0000	1.4142	0.4525
3	60	80	80	77.05	0.2486	0.0000	2.8284	0.4638
3	60	80	100	76.64	0.2514	0.0000	2.2361	0.4623
3	60	80	120	77.05	0.2469	0.0000	2.2361	0.4591
3	60	100	60	77.26	0.2463	0.0000	2.8284	0.4614
3	60	100	80	77.19	0.2465	0.0000	2.8284	0.4608
3	60	100	100	77.60	0.2419	0.0000	2.0000	0.4570
3	60	100	120	77.14	0.2462	0.0000	2.2361	0.4592
3	60	120	60	77.41	0.2447	0.0000	2.2361	0.4602
3	60	120	80	76.71	0.2508	0.0000	2.0000	0.4619
3	60	120	100	76.60	0.2512	0.0000	1.4142	0.4609
3	60	120	120	76.17	0.2585	0.0000	2.8284	0.4713
3	80	60	60	78.23	0.2341	0.0000	2.2361	0.4504
3	80	60	80	77.48	0.2426	0.0000	2.8284	0.4572
3	80	60	100	77.37	0.2448	0.0000	2.2361	0.4599
3	80	60	120	76.73	0.2530	0.0000	2.8284	0.4678
3	80	80	60	78.41	0.2347	0.0000	2.2361	0.4544
3	80	80	80	77.14	0.2454	0.0000	2.0000	0.4574
3	80	80	100	76.92	0.2501	0.0000	2.8284	0.4647
3	80	80	120	77.62	0.2422	0.0000	2.8284	0.4586

ตารางที่ 4.11 ผลของ Hidden Layer ต่อแบบจำลอง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	60	60	60	76.49	0.2545	0.0000	2.8284	0.4670
3	80	100	60	77.87	0.2390	0.0000	2.8284	0.4561
3	80	100	80	77.32	0.2465	0.0000	2.2361	0.4627
3	80	100	100	77.23	0.2458	0.0000	2.2361	0.4597
3	80	100	120	76.51	0.2532	0.0000	2.0000	0.4640
3	80	120	60	77.48	0.2438	0.0000	1.4142	0.4590
3	80	120	80	77.55	0.2429	0.0000	1.4142	0.4582
3	80	120	100	76.26	0.2560	0.0000	1.4142	0.4656
3	80	120	120	77.01	0.2485	0.0000	2.2361	0.4619
3	100	60	60	78.57	0.2319	0.0000	2.0000	0.4510
3	100	60	80	78.03	0.2387	0.0000	2.0000	0.4570
3	100	60	100	77.39	0.2443	0.0000	2.8284	0.4599
3	100	60	120	77.76	0.2396	0.0000	2.8284	0.4555
3	100	80	60	77.48	0.2440	0.0000	1.4142	0.4594
3	100	80	80	76.26	0.2566	0.0000	2.0000	0.4670
3	100	80	100	77.03	0.2494	0.0000	2.2361	0.4644
3	100	80	120	77.48	0.2438	0.0000	2.0000	0.4593
3	100	100	60	77.51	0.2434	0.0000	1.4142	0.4587
3	100	100	80	77.44	0.2440	0.0000	2.0000	0.4591
3	100	100	100	76.69	0.2518	0.0000	2.8284	0.4648
3	100	100	120	76.64	0.2542	0.0000	2.8284	0.4688
3	100	120	60	77.53	0.2432	0.0000	2.8284	0.4591
3	100	120	80	77.30	0.2459	0.0000	2.8284	0.4618
3	100	120	100	76.76	0.2506	0.0000	2.2361	0.4623

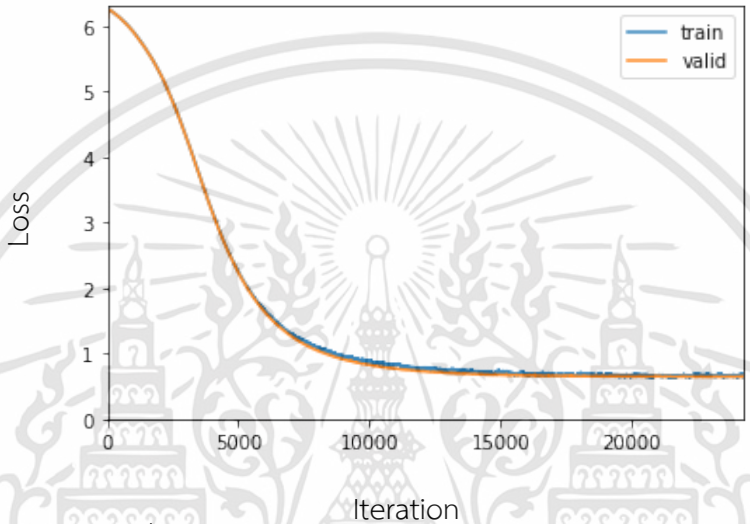
ตารางที่ 4.11 ผลของ Hidden Layer ต่อแบบจำลอง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	100	120	120	76.37	0.2541	0.0000	2.0000	0.4635
3	120	60	60	78.00	0.2388	0.0000	2.0000	0.4567
3	120	60	80	77.69	0.2413	0.0000	2.2361	0.4574
3	120	60	100	77.28	0.2463	0.0000	2.2361	0.4616
3	120	60	120	77.17	0.2462	0.0000	2.2361	0.4597
3	120	80	60	77.66	0.2428	0.0000	2.2361	0.4600
3	120	80	80	76.80	0.2518	0.0000	2.0000	0.4655
3	120	80	100	77.80	0.2401	0.0000	2.2361	0.4565
3	120	80	120	76.64	0.2525	0.0000	2.2361	0.4649
3	120	100	60	77.51	0.2429	0.0000	2.2361	0.4578
3	120	100	80	77.30	0.2455	0.0000	2.0000	0.4601
3	120	100	100	76.76	0.2508	0.0000	2.2361	0.4628
3	120	100	120	76.10	0.2591	0.0000	2.2361	0.4700
3	120	120	60	76.92	0.2494	0.0000	1.4142	0.4621
3	120	120	80	76.94	0.2492	0.0000	1.4142	0.4620
3	120	120	100	76.73	0.2497	0.0000	2.2361	0.4603
3	120	120	120	76.46	0.2549	0.0000	2.8284	0.4677

5) ประเมินผลแบบจำลอง

จากการใช้แบบจำลองของชุดข้อมูลของขนาดพื้นที่ 6 x 6 เมตร ในอัตราส่วน 0.3 ทั้งหมด 441 ตำแหน่ง ที่มีเครื่องส่ง 12 ตัว ผ่านการฝึกฝนแบบจำลองด้วยขนาด Batch size เท่ากับ 640 ใช้อัตราการเรียนรู้เท่ากับ 0.0001 มีจำนวนการฝึกฝนเท่ากับ 900 รอบ ใช้จำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 100 60 และ 60 ตามลำดับ ผลลัพธ์ที่ได้คือแบบจำลองที่มีความแม่นยำ 78.57% โดยมีระยะคลาดเคลื่อนเฉลี่ย 0.2319 เมตร

ระยะคลาดเคลื่อนต่ำสุด 0 เมตร ระยะคลาดเคลื่อนสูงสุด 2 เมตร และมีส่วนเบี่ยงเบนมาตรฐานเท่ากับ 0.451 เมตร แสดงลักษณะการลดลงของค่า Loss ระหว่างกราฟ train สำหรับข้อมูลที่ใช้ในการฝึกฝน และกราฟ valid สำหรับข้อมูลที่ใช้ในการทดสอบ เมื่อแกนตั้งคือค่า Loss และแกนนอนคือจำนวน Iteration ดังรูปที่ 4.1 ซึ่งไม่มีการเกิด Underfitting หรือ Overfitting



รูปที่ 4.1 กราฟ Model Loss ของการฝึกฝนแบบจำลอง

4.1.5 การระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาทภายในห้องทดลอง

จากผลลัพธ์ของการทดสอบด้วยแบบจำลอง จึงเลือกใช้อัตราส่วนเท่ากับ 0.4 เมตร แล้วจึงลดจำนวนเครื่องส่งเหลือเพียง 4 ตัว เพราะการใช้เครื่องส่งที่มากไปเป็นการเพิ่มการติดตั้ง ซึ่งการใช้งานเครื่องส่งที่มากเกินไปไม่เหมาะกับการนำมาใช้งานจริงด้วยค่าใช้จ่ายที่มากเกินจำเป็น ภายในห้องทดลองมีขนาด 3.5 x 4.0 เมตร เมื่อคำนวณตามอัตราส่วนที่กำหนด ทำให้มีจุดในการระบุตำแหน่งทั้งหมด 72 ตำแหน่ง ทำการเก็บข้อมูล RSSI ตำแหน่งละ 200 samples ทำให้ได้ชุดข้อมูลที่มี 14,400 samples ถูกแบ่งด้วยอัตราส่วน 80:20 สำหรับฝึกฝนและทดสอบ โดยจะถูกนำมาทดสอบหาค่า Hyperparameter ที่เหมาะสมต่อไป

1) Batch Size

ในการทดสอบนี้จะใช้ค่า Hyperparameter คือจำนวน Hidden Layer เท่ากับ 3 จำนวนเซลล์ประสาทเท่ากับ 60 60 และ 60 ตามลำดับ มีจำนวนการฝึกฝนที่ 400 รอบ และใช้อัตราการเรียนรู้เท่ากับ 0.0001 แล้วทำการทดสอบ Batch Size ที่ขนาด 32 64 128 256 384 512 640 768 896 และ 1024 ให้ผลลัพธ์ดังตารางที่ 4.12 โดยขนาด Batch Size ที่เหมาะสมที่จะนำมาใช้งานคือ 256

ตารางที่ 4.12 ผลของ Batch Size ต่อโมเดลห้องทดลอง

Batch Size	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
32	90.87	0.1035	0.0000	2.9120	0.3801
64	92.26	0.0847	0.0000	2.9120	0.3400
128	92.40	0.0842	0.0000	3.0463	0.3422
256	92.47	0.0879	0.0000	2.8284	0.3538
384	92.50	0.0859	0.0000	3.0463	0.3509
512	92.40	0.0840	0.0000	2.9120	0.3398
640	92.50	0.0834	0.0000	2.9120	0.3379
768	91.94	0.0945	0.0000	2.9120	0.3685
896	91.88	0.0973	0.0000	3.0463	0.3805
1024	91.25	0.0981	0.0000	2.9120	0.3681

2) จำนวนการฝึกฝน

ในการทดสอบนี้จะเป็นการทดสอบที่ต่อจากข้อ 1) โดยใช้ค่า Hyperparameter เช่นเดียวกัน แต่จะใช้ขนาด Batch Size เท่ากับ 256 ที่ได้จากการทดสอบมาก่อนหน้านี้ ส่วนจำนวนการฝึกฝนจะทดสอบที่ 100 – 1000 รอบ ให้ผลลัพธ์ดังตารางที่ 4.13 จำนวนการฝึกฝนที่เหมาะสมคือ 400 - 800 รอบ

ตารางที่ 4.13 ผลของจำนวนการฝึกฝนต่อโมเดลห้องทดลอง

จำนวนการฝึกฝน (รอบ)	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
100	86.25	0.1498	0.0000	3.0463	0.4366
200	90.24	0.1084	0.0000	3.0463	0.3860
300	91.53	0.0947	0.0000	3.0463	0.3621
400	92.12	0.0901	0.0000	3.0463	0.3572
500	92.60	0.0854	0.0000	3.0463	0.3488
600	92.71	0.0855	0.0000	3.0463	0.3520
700	93.02	0.0833	0.0000	3.0463	0.3495
800	93.40	0.0787	0.0000	2.8284	0.3400
900	93.33	0.0803	0.0000	2.9120	0.3440
1000	93.30	0.0799	0.0000	2.9120	0.3412

3) อัตราการเรียนรู้

ในการทดสอบนี้จะเป็นการทดสอบที่ต่อจากข้อ 2) โดยใช้ค่า Hyperparameter เช่นเดียวกัน ซึ่งก็คือมีจำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 60 60 และ 60 ตามลำดับ และมีขนาด Batch Size เท่ากับ 640 แต่จะใช้จำนวนการฝึกฝนที่ 600 รอบ ที่ได้มาจากการทดสอบก่อนหน้านี้ ส่วนอัตราการเรียนรู้จะทดสอบที่ 0.001 0.0009 0.0008 0.0007 0.0006 0.0005 0.0004 0.0003 0.0002 และ 0.0001 ให้ผลลัพธ์ดังตารางที่ 4.14 อัตราการเรียนรู้ที่เหมาะสมคือ 0.0001 – 0.0002

ตารางที่ 4.14 ผลของอัตราการเรียนรู้ต่อโมเดลห้องทดลอง

อัตราการเรียนรู้	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
0.0010	93.47	0.0778	0.0000	3.1241	0.3374
0.0009	93.75	0.0741	0.0000	2.8844	0.3282
0.0008	93.65	0.0772	0.0000	3.4409	0.3419
0.0007	93.40	0.0793	0.0000	3.9598	0.3482
0.0006	93.51	0.0755	0.0000	3.1241	0.3316
0.0005	93.65	0.0765	0.0000	2.9120	0.3337
0.0004	93.78	0.0716	0.0000	4.0000	0.3191
0.0003	93.54	0.0715	0.0000	3.1241	0.3116
0.0002	93.61	0.0687	0.0000	2.8000	0.3041
0.0001	93.16	0.0725	0.0000	2.5612	0.3083

4) Hidden Layer

ในการทดสอบนี้จะเป็นการทดสอบที่ต่อจากข้อ 3) โดยใช้ค่า Hyperparameter เช่นเดียวกัน ซึ่งก็คือใช้ขนาด Batch Size เท่ากับ 256 มีจำนวนการฝึกฝนที่ 600 รอบ แต่จะใช้อัตราการเรียนรู้เท่ากับ 0.0002 ที่ได้มาจากการทดสอบก่อนหน้านี้ ส่วน Hidden Layer ได้ทำการทดสอบจำนวนเซลล์ประสาทตั้งแต่ 60 – 120 ให้ผลลัพธ์ดังตารางที่ 4.15 ที่เซลล์ประสาทเท่ากับ 60 120 และ 60 ตามลำดับนั้นให้ความแม่นยำมากที่สุด

ตารางที่ 4.15 ผลของ Hidden Layer ต่อโมเดลห้องทดลอง

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	60	60	60	93.89	0.0688	0.0000	2.9120	0.3154
3	60	60	80	93.47	0.0735	0.0000	2.9120	0.3198
3	60	60	100	93.65	0.0716	0.0000	2.8844	0.3194
3	60	60	120	93.75	0.0708	0.0000	2.8284	0.3144
3	60	80	60	93.65	0.0757	0.0000	2.9120	0.3322
3	60	80	80	93.75	0.0709	0.0000	3.3941	0.3156
3	60	80	100	93.68	0.0726	0.0000	2.9120	0.3205
3	60	80	120	93.54	0.0738	0.0000	2.9120	0.3230
3	60	100	60	93.61	0.0714	0.0000	2.8000	0.3161
3	60	100	80	93.82	0.0694	0.0000	2.8284	0.3108
3	60	100	100	93.96	0.0681	0.0000	3.1241	0.3083
3	60	100	120	93.51	0.0746	0.0000	3.1241	0.3274
3	60	120	60	93.72	0.0669	0.0000	2.8284	0.2970
3	60	120	80	93.75	0.0710	0.0000	2.9120	0.3170
3	60	120	100	94.03	0.0681	0.0000	2.6833	0.3074
3	60	120	120	93.51	0.0727	0.0000	2.9120	0.3165
3	80	60	60	93.96	0.0692	0.0000	3.1241	0.3159
3	80	60	80	93.68	0.0720	0.0000	2.8284	0.3188
3	80	60	100	93.40	0.0727	0.0000	2.9120	0.3190
3	80	60	120	93.61	0.0744	0.0000	2.9120	0.3287
3	80	80	60	93.58	0.0751	0.0000	3.1241	0.3314
3	80	80	80	93.75	0.0718	0.0000	3.0463	0.3218
3	80	80	100	93.61	0.0707	0.0000	2.8000	0.3111
3	80	80	120	93.78	0.0726	0.0000	2.9120	0.3246

ตารางที่ 4.15 ผลของ Hidden Layer ต่อโมเดลห้องทดลอง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	80	100	60	93.44	0.0750	0.0000	2.8000	0.3230
3	80	100	80	93.75	0.0712	0.0000	2.8284	0.3166
3	80	100	100	93.99	0.0688	0.0000	2.8284	0.3134
3	80	100	120	93.44	0.0746	0.0000	3.1241	0.3231
3	80	120	60	93.85	0.0703	0.0000	3.1241	0.3168
3	80	120	80	93.89	0.0698	0.0000	2.8284	0.3129
3	80	120	100	93.61	0.0722	0.0000	2.8000	0.3166
3	80	120	120	93.65	0.0719	0.0000	2.6833	0.3164
3	100	60	60	93.85	0.0715	0.0000	2.8284	0.3164
3	100	60	80	93.51	0.0740	0.0000	3.0463	0.3229
3	100	60	100	93.85	0.0707	0.0000	2.8284	0.3171
3	100	60	120	93.37	0.0731	0.0000	3.4409	0.3188
3	100	80	60	93.58	0.0743	0.0000	3.2249	0.3279
3	100	80	80	93.47	0.0763	0.0000	3.1241	0.3331
3	100	80	100	93.89	0.0711	0.0000	2.6833	0.3179
3	100	80	120	93.72	0.0695	0.0000	2.8284	0.3090
3	100	100	60	93.65	0.0729	0.0000	3.9598	0.3287
3	100	100	80	93.68	0.0688	0.0000	2.9120	0.3051
3	100	100	100	93.54	0.0712	0.0000	2.8284	0.3116
3	100	100	120	93.51	0.0736	0.0000	2.8284	0.3230
3	100	120	60	93.85	0.0685	0.0000	2.8000	0.3064
3	100	120	80	93.75	0.0717	0.0000	2.8000	0.3189
3	100	120	100	93.85	0.0720	0.0000	2.9120	0.3254
3	100	120	120	93.58	0.0740	0.0000	2.9120	0.3240

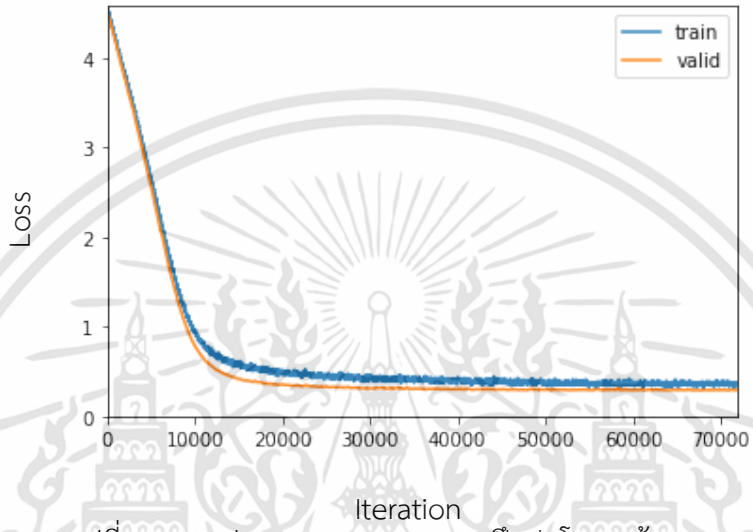
ตารางที่ 4.15 ผลของ Hidden Layer ต่อโมเดลห้องทดลอง (ต่อ)

จำนวน Hidden Layer	จำนวนเซลล์ประสาท			ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
	Layer 1	Layer 2	Layer 3		เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
3	120	60	60	93.82	0.0694	0.0000	2.9120	0.3085
3	120	60	80	93.72	0.0751	0.0000	2.9120	0.3344
3	120	60	100	93.58	0.0732	0.0000	2.8000	0.3213
3	120	60	120	93.58	0.0764	0.0000	2.9120	0.3354
3	120	80	60	93.54	0.0767	0.0000	3.1241	0.3391
3	120	80	80	94.06	0.0695	0.0000	2.8284	0.3145
3	120	80	100	93.47	0.0744	0.0000	2.8284	0.3249
3	120	80	120	93.68	0.0725	0.0000	2.8284	0.3216
3	120	100	60	93.37	0.0773	0.0000	2.9120	0.3324
3	120	100	80	93.68	0.0721	0.0000	2.8844	0.3224
3	120	100	100	93.85	0.0694	0.0000	2.8000	0.3105
3	120	100	120	93.47	0.0741	0.0000	2.9120	0.3230
3	120	120	60	93.78	0.0712	0.0000	3.9598	0.3211
3	120	120	80	93.61	0.0744	0.0000	2.8284	0.3271
3	120	120	100	93.58	0.0755	0.0000	3.1241	0.3346
3	120	120	120	93.92	0.0688	0.0000	2.8284	0.3123

5) ประเมินผลแบบจำลอง

จากการทดสอบในห้องทดลองขนาดพื้นที่ 3.5 x 4 เมตร ในอัตราส่วน 0.4 เมตร ทั้งหมด 72 ตำแหน่ง ที่มีเครื่องส่ง 4 ตัว ผ่านการฝึกฝนแบบจำลองด้วยขนาด Batch size เท่ากับ 256 ใช้อัตราการเรียนรู้เท่ากับ 0.0002 มีจำนวนการฝึกฝนเท่ากับ 600 รอบ ใช้จำนวน Hidden Layer เท่ากับ 3 มีจำนวนเซลล์ประสาทเท่ากับ 60 120 และ 60 ตามลำดับ ผลลัพธ์ที่ได้คือแบบจำลองที่มีความแม่นยำ 93.72% โดยมีระยะคลาดเคลื่อนเฉลี่ย 0.0669 เมตร ระยะคลาดเคลื่อนต่ำสุด 0 เมตร ระยะคลาดเคลื่อนสูงสุด 2.8284 เมตร และมีส่วนเบี่ยงเบนมาตรฐาน

เท่ากับ 0.2970 เมตร แสดงลักษณะการลดลงของค่า Loss ระหว่างกราฟ train สำหรับข้อมูลที่ใช้ในการฝึกฝน และกราฟ valid สำหรับข้อมูลที่ใช้ในการทดสอบ เมื่อแกนตั้งคือค่า Loss และแกนนอนคือจำนวน Iteration ดังรูปที่ 4.2 ซึ่งไม่มีการเกิด Underfitting หรือ Overfitting



รูปที่ 4.2 กราฟ Model Loss ของการฝึกฝนโมเดลห้องทดลอง

4.2 การทดสอบหาพารามิเตอร์สำหรับแบบจำลอง Path Loss

ในการทดสอบนี้จะเก็บค่า RSSI จากเครื่องส่งไมโครคอนโทรลเลอร์ ESP32 ที่ส่งมายังเครื่องรับโทรศัพท์เคลื่อนที่ด้วยโปรแกรมประยุกต์ที่ออกแบบมาสำหรับการบันทึกค่า โดยทำการวัด RSSI ที่ระยะ 0.2 – 15 เมตร ระยะละ 30 Samples ได้ผลลัพธ์ดังตารางที่ 4.16 เพื่อนำไปใช้สำหรับวิเคราะห์หาพารามิเตอร์ของแบบจำลอง Path Loss ซึ่งจะนำไปใช้สำหรับการสร้างแบบจำลองชุดข้อมูล

ตารางที่ 4.16 ผลการทดสอบ RSSI ที่ระยะ 0.2 – 15 เมตร

ระยะทาง (เมตร)	RSSI (dBm)				
	เฉลี่ย	ฐานนิยม	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
0.2	-32.2333	-32	-34	-31	0.8172
0.4	-42.3333	-43	-43	-38	1.2685
0.6	-42.3000	-43	-43	-41	0.7022

ตารางที่ 4.16 ผลการทดสอบ RSSI ที่ระยะ 0.2 – 15 เมตร (ต่อ)

ระยะทาง (เมตร)	RSSI (dBm)				
	เฉลี่ย	ฐานนิยม	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
0.8	-51.1000	-49	-55	-49	1.9888
1.0	-57.4333	-58	-59	-53	1.9061
1.2	-64.3333	-62	-68	-61	2.5371
1.4	-69.0667	-67	-73	-64	2.4202
1.6	-66.8000	-67	-70	-65	1.6897
1.8	-65.5333	-69	-70	-61	3.1265
2.0	-68.6000	-68	-71	-64	1.9405
2.2	-66.6333	-67	-68	-65	1.0334
2.4	-69.9667	-69	-72	-68	1.2172
2.6	-71.9000	-74	-78	-67	3.3151
2.8	-72.8667	-72	-76	-67	2.0297
3.0	-73.9333	-74	-79	-71	2.1804
3.2	-75.6333	-75	-77	-74	1.0334
3.4	-77.0667	-77	-79	-75	1.0483
3.6	-68.1333	-68	-74	-65	2.0466
3.8	-71.7333	-71	-76	-69	2.3034
4.0	-67.1333	-67	-71	-66	1.3578
4.2	-76.4333	-75	-80	-73	2.3146
4.4	-72.2333	-72	-76	-71	1.4308
4.6	-70.3000	-70	-74	-68	1.3684
4.8	-72.4333	-73	-74	-68	1.4308
5.0	-70.2000	-72	-74	-65	2.5515
5.2	-75.4000	-76	-79	-70	2.2530
5.4	-73.5667	-74	-78	-69	2.2846
5.6	-70.1667	-70	-73	-66	1.6206

ตารางที่ 4.16 ผลการทดสอบ RSSI ที่ระยะ 0.2 – 15 เมตร (ต่อ)

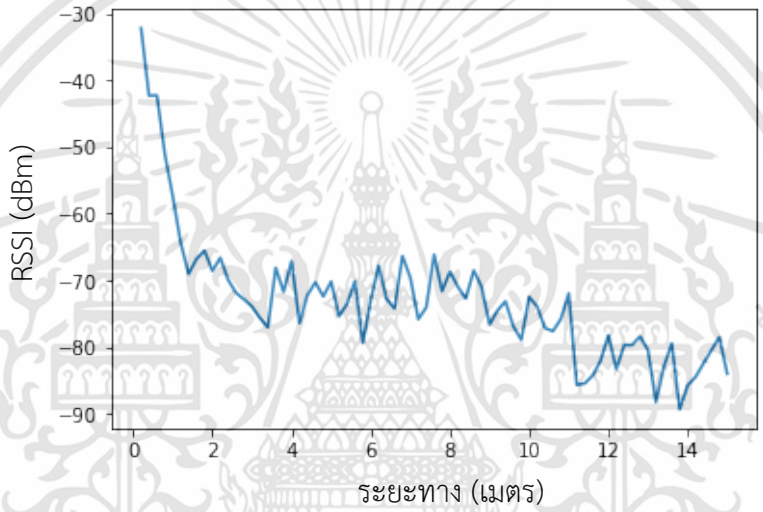
ระยะทาง (เมตร)	RSSI (dBm)				
	เฉลี่ย	ฐานนิยม	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
5.8	-79.4000	-80	-84	-73	3.0581
6.0	-72.9333	-73	-80	-68	3.5129
6.2	-67.8000	-67	-73	-66	1.9547
6.4	-72.8000	-71	-85	-68	4.4675
6.6	-74.2333	-74	-79	-69	2.5955
6.8	-66.3667	-65	-72	-64	1.8473
7.0	-69.6333	-69	-74	-68	1.2172
7.2	-75.8333	-78	-79	-71	2.3057
7.4	-74.0667	-73	-77	-72	1.3374
7.6	-66.1667	-66	-68	-64	1.1769
7.8	-71.6333	-71	-76	-65	2.5255
8.0	-68.7000	-67	-72	-66	2.0536
8.2	-71.0000	-71	-73	-66	1.2034
8.4	-72.7333	-72	-79	-69	2.3479
8.6	-68.5333	-67	-79	-64	4.0661
8.8	-71.0333	-71	-75	-69	1.4499
9.0	-76.6667	-79	-79	-74	2.0734
9.2	-74.5333	-75	-77	-71	1.9780
9.4	-73.1667	-73	-75	-72	0.8743
9.6	-76.9000	-77	-82	-73	2.5100
9.8	-78.9333	-80	-81	-74	1.6802
10.0	-72.5000	-71	-80	-67	3.6742
10.2	-73.9333	-74	-81	-71	2.1804
10.4	-77.1667	-77	-80	-76	1.0199
10.6	-77.6333	-77	-81	-76	1.3515

ตารางที่ 4.16 ผลการทดสอบ RSSI ที่ระยะ 0.2 – 15 เมตร (ต่อ)

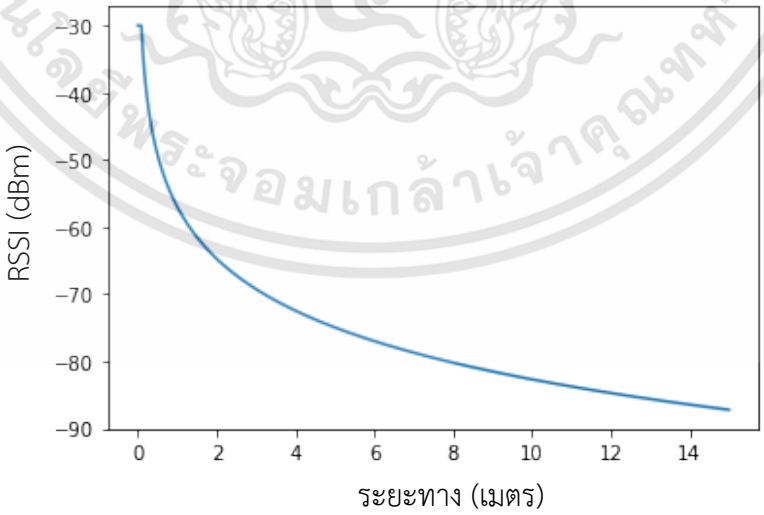
ระยะทาง (เมตร)	RSSI (dBm)				
	เฉลี่ย	ฐานนิยม	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
10.8	-75.8000	-76	-77	-74	0.8052
11.0	-71.9667	-72	-73	-71	0.4138
11.2	-85.6667	-85	-91	-79	3.1550
11.4	-85.4667	-88	-89	-80	2.9094
11.6	-84.3000	-85	-87	-81	1.4179
11.8	-82.1000	-83	-84	-76	1.6049
12.0	-78.2333	-79	-81	-74	1.6955
12.2	-83.3333	-84	-86	-75	1.9535
12.4	-79.7000	-79	-83	-79	0.9523
12.6	-79.7000	-80	-84	-75	2.3216
12.8	-78.4000	-78	-81	-77	1.0372
13.0	-80.4667	-79	-91	-74	2.9680
13.2	-88.2667	-88	-90	-87	0.6915
13.4	-82.9667	-82	-86	-82	1.2452
13.6	-79.4667	-80	-81	-78	0.7761
13.8	-89.3333	-89	-92	-83	1.5610
14.0	-85.7667	-85	-87	-83	1.1351
14.2	-84.5000	-85	-87	-83	1.0086
14.4	-82.4667	-83	-84	-81	0.8193
14.6	-80.5000	-82	-83	-76	2.1294
14.8	-78.5333	-79	-79	-76	0.8193
15.0	-84.0000	-84	-87	-83	1.0171

เมื่อนำผลลัพธ์เฉลี่ยที่ได้นำมาวาดกราฟความสัมพันธ์ระหว่างค่า RSSI กับระยะทาง จะได้กราฟดังรูปที่ 4.3 จะเห็นได้ว่าที่ระยะ 1.6 – 15 เมตร ค่า RSSI มีการแกว่งที่สูงมาก และยาก

ต่อการนำมาใช้สำหรับบอกระยะทางด้วยฟังก์ชันของแบบจำลอง Path Loss เพราะกราฟไม่เป็นเชิงเส้น จึงได้นำความสัมพันธ์นี้นำมาคำนวณหาค่าเฉลี่ยของพารามิเตอร์ η ในสมการที่ (2.6) เมื่อเฉลี่ยค่า η จะได้เท่ากับ 2.5911 เมื่อกำหนดให้มีการอ้างอิงที่ระยะ 1 เมตร คือ $RSSI_{d_0}$ เท่ากับ -57 dBm และ d_0 เท่ากับ 1 เมตร เมื่อได้พารามิเตอร์ครบแล้วจะถูกนำมาเขียนเป็นฟังก์ชัน $RSSI$ ด้วยระยะทาง d ตามสมการที่ (2.6) เมื่อพล็อตกราฟจะได้ความสัมพันธ์ระหว่างค่า RSSI และระยะทางแบบเป็นเชิงเส้นดังรูปที่ 4.4 ซึ่งสามารถนำค่าพารามิเตอร์นี้ไปใช้สำหรับการทำแบบจำลองชุดข้อมูลต่อไป



รูปที่ 4.3 กราฟความสัมพันธ์ระหว่างค่า RSSI กับระยะทางที่เก็บค่าได้

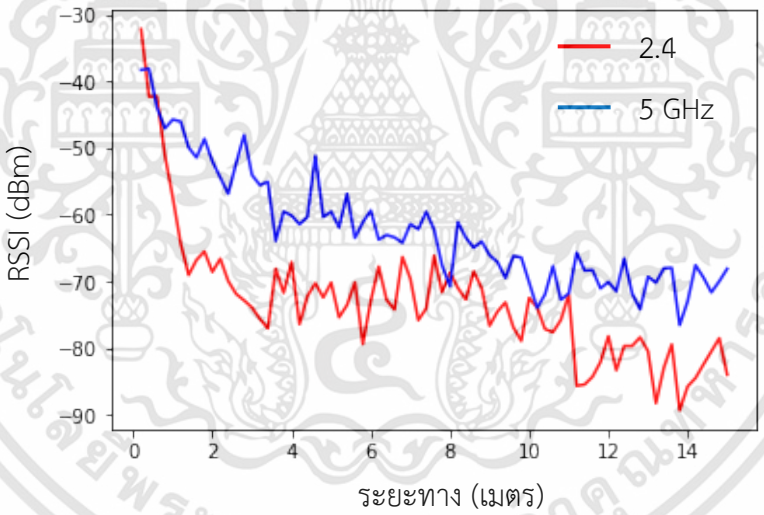


รูปที่ 4.4 กราฟความสัมพันธ์ระหว่างค่า RSSI กับระยะทางจากการคำนวณหาพารามิเตอร์

4.3 การทดสอบการปรับปรุงระบบการระบุตำแหน่ง

4.3.1 การใช้งานเครื่องส่งความถี่ 5 GHz

จากการทดสอบโดยเก็บข้อมูล RSSI ของเครื่องส่งความถี่ 2.4 GHz และ 5 GHz ที่ระยะ 0 – 15 เมตร ให้ผลลัพธ์ดังรูปที่ 4.5 ปรากฏว่าเครื่องส่งความถี่ 2.4 GHz มีค่า RSSI ที่เปลี่ยนแปลงอย่างรวดเร็วในช่วง 0 – 4 เมตร และเปลี่ยนแปลงอย่างช้า ๆ ในระยะ 4 – 15 เมตร และเครื่องส่งความถี่ 5 GHz มีค่า RSSI ที่เปลี่ยนแปลงอย่างช้า ๆ ในช่วงระยะ 0 – 15 เมตร สรุปได้ว่า เครื่องส่งความถี่ 5 GHz ไม่เหมาะกับการนำมาใช้ระบุตำแหน่ง เพราะลักษณะของค่า RSSI ที่เปลี่ยนแปลงอย่างช้า ๆ ตามระยะทาง ส่งผลให้ระบบระบุตำแหน่งมีความคลาดเคลื่อนสูง เมื่อมีสัญญาณรบกวนมากระทำกับสัญญาณจากเครื่องส่ง โดยเครื่องส่งความถี่ 2.4 GHz มีการเปลี่ยนแปลงของค่า RSSI ตามระยะทางที่สูง ทำให้ทนต่อสัญญาณรบกวนที่มากระทำกับสัญญาณจากเครื่องส่ง



รูปที่ 4.5 กราฟความสัมพันธ์ของค่า RSSI กับระยะทางของเครื่องส่งความถี่ 2.4 GHz และ 5 GHz

4.3.2 การใช้งานระบบจำแนกข้อมูลอนุกรมด้วยการเรียนรู้เชิงลึก

จากการทดสอบด้วยระบบจำแนกข้อมูลอนุกรมด้วยการเรียนรู้เชิงลึกด้วยจำนวนลำดับอนุกรม 5 10 15 และ 20 ต่อ sample โดยแบ่งชุดข้อมูลด้วยอัตราส่วน 80:20 สำหรับฝึกฝนและทดสอบ ใช้จำนวนการฝึกฝน 200 รอบ Batch Size เท่ากับจำนวนลำดับอนุกรม จำนวนเซลล์

ประสาทเท่ากับ 100 ให้ผลลัพธ์เมื่อเทียบกับวิธีแบบปกติ ดังตารางที่ 4.17 สรุปได้ว่าการใช้วิธีจำแนกข้อมูลอนุกรมด้วยการเรียนรู้เชิงลึกไม่ได้ทำให้ผลลัพธ์ดีขึ้นจากวิธีปกติ

ตารางที่ 4.17 ผลการทดสอบระบบจำแนกข้อมูลอนุกรมด้วยการเรียนรู้เชิงลึก

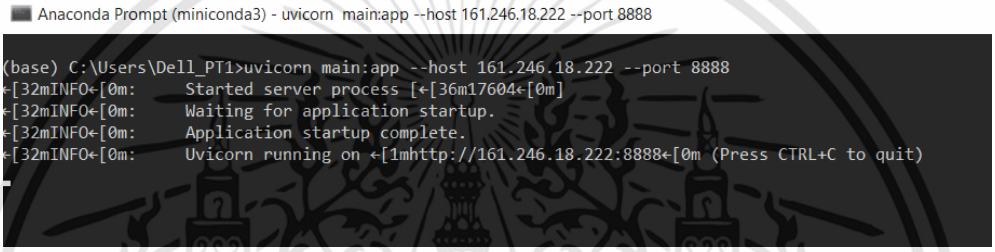
อัตราส่วน (เมตร)	ชนิด	จำนวนลำดับอนุกรม	ความแม่นยำ (%)	ระยะคลาดเคลื่อน (เมตร)			
				เฉลี่ย	น้อยสุด	มากสุด	ส่วนเบี่ยงเบนมาตรฐาน
0.3	ปกติ	-	92.25	0.0991	0.0000	3.6125	0.417
	อนุกรม	5	79.90	0.2292	0.0000	4.0361	0.5787
		10	82.96	0.1878	0.0000	3.4205	0.5226
		15	83.88	0.1857	0.0000	3.6125	0.5337
		20	84.81	0.1278	0.0000	2.7166	0.4045
0.4	ปกติ	-	93.43	0.0701	0.0000	3.1241	0.33
	อนุกรม	5	86.24	0.111	0.0000	2.6833	0.3382
		10	88.62	0.1088	0.0000	2.3324	0.3765
		15	89.28	0.1041	0.0000	2.8284	0.3927
		20	89.94	0.1178	0.0000	2.6833	0.4122
0.5	ปกติ	-	95.59	0.0589	0.0000	2.8284	0.3062
	อนุกรม	5	92.59	0.1121	0.0000	2.6926	0.4247
		10	91.85	0.0962	0.0000	2.8284	0.362
		15	91.11	0.1402	0.0000	2.5495	0.4871
		20	92.59	0.0975	0.0000	2.6926	0.4026

4.3.3 การประยุกต์ใช้งานภายในอาคาร

กำหนดพื้นที่ขนาด 1.5 x 34.0 เมตร บริเวณทางเดินอาคาร ใช้จำนวนเครื่องส่ง 6 ตัว เลือกใช้อัตราส่วน 0.5 เมตร ซึ่งมีพิกัดทั้งหมด 134 ตำแหน่ง ฝึกฝนด้วย Hyperparameter ที่เหมาะสมจากผลการทดสอบในห้องทดลอง จากการทดสอบและทำให้ได้มาของระบบระบุตำแหน่งภายในอาคารที่มีความแม่นยำ 93.00% มีระยะคลาดเคลื่อนเฉลี่ย 0.2562 น้อยสุด 0 เมตร มากสุด 21 เมตร และมีส่วนเบี่ยงเบนมาตรฐาน 1.4333 เมตร

4.4 การทดสอบการทำงานของเซิร์ฟเวอร์

โปรแกรมเซิร์ฟเวอร์จะถูกรันบน miniconda เนื่องจากมีความเข้ากันได้กับชุดคำสั่งของ fastai บนระบบปฏิบัติการ Windows มากที่สุด เมื่อรันโปรแกรมเซิร์ฟเวอร์ผ่านที่อยู่ ip 161.246.18.222 port 8888 จะได้ผลลัพธ์ดังรูปที่ 4.6 ซึ่งสามารถตรวจสอบ API ได้จาก http://161.246.18.222:8888/docs จะปรากฏเอกสารคำสั่งการใช้งาน API นี้ ดังรูปที่ 4.7 และเมื่อทดสอบการทำงานโดยการส่งข้อมูล RSSI ไปยังเซิร์ฟเวอร์ผ่าน API นี้จะได้สำเร็จ จะได้ผลลัพธ์ดังรูปที่ 4.8



รูปที่ 4.6 โปรแกรมเซิร์ฟเวอร์บน miniconda

รูปที่ 4.7 เอกสารการใช้งาน API ของเซิร์ฟเวอร์ประมวลผล

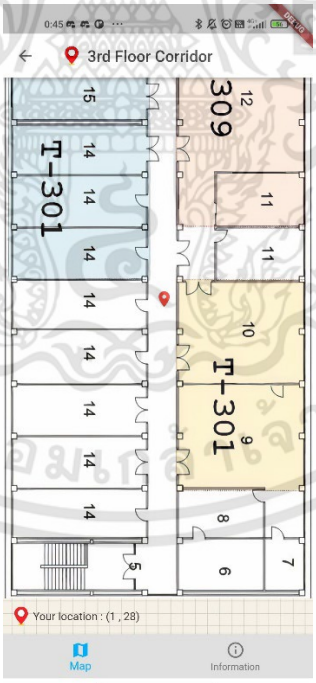
```
[5]
from google.colab import drive
drive.mount('/content/drive')

path = '/content/drive/MyDrive/Engineer/Project/'
map = pd.read_csv(path+'new_0.3_20.csv')
num = 19000
df = map.loc[num]
#print(df)
data ={}
for i in range(12):
    data["{}".format(df.index[i])] = df[i]
data = json.dumps(data)
#print(data)

Drive already mounted at /content/drive; to attempt to forcibly remount,

[8]
##### Send RSSI data to sever & Get the predicted position #####
b = requests.post('http://161.246.18.222:8888/rssi/', data=data)
print(b)
print(b.json())

<Response [200]>
{'x': 10, 'y': 3}

##### Get the predicted position #####
y = requests.get('http://161.246.18.222:8888/pos/')
print(y.json())

{'x': 10, 'y': 3}
```

รูปที่ 4.8 ตัวอย่างผลการทดสอบการทำงานของเซิร์ฟเวอร์ประมวลผล

4.5 การทดสอบการทำงานของโปรแกรมประยุกต์

ทำการทดสอบการทำงานของโปรแกรมประยุกต์ โดยให้โปรแกรมประยุกต์ส่งค่า RSSI ไปยังเซิร์ฟเวอร์จากนั้นเซิร์ฟเวอร์จะทำการประมวลผล และส่งค่าที่เป็นพิกัด (x, y) กลับมาแสดงผลบนโปรแกรมประยุกต์ ผลลัพธ์แสดงดังรูปที่ 4.9 และ 4.10

รูปที่ 4.9 โปรแกรมประยุกต์แสดงพิกัด (x, y) ที่ได้จากการประมวลผล (Co-working Space)



รูปที่ 4.10 โปรแกรมประยุกต์แสดงพิกัด (x, y) ที่ได้จากการประมวลผล (3rd Floor Corridor)

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้มีวัตถุประสงค์เพื่อออกแบบระบบระบุตำแหน่งภายในอาคารโดยใช้ค่า RSSI ประมวลผลด้วยอัลกอริทึมการเรียนรู้เชิงลึกเพื่อให้สามารถระบุตำแหน่งได้อย่างแม่นยำ และพัฒนาเป็นโปรแกรมประยุกต์ที่ใช้ระบุตำแหน่งบนโทรศัพท์เคลื่อนที่ ซึ่งระบบระบุตำแหน่งภายในอาคารที่นำเสนอแบ่งออกเป็น 3 ส่วนหลักๆ คือ 1) ส่วนนำเข้าข้อมูล ซึ่งจะรับค่า RSSI และส่งต่อไปยังส่วนประมวลผล 2) ส่วนประมวลผลข้อมูล จะรับข้อมูลค่า RSSI จากส่วนนำเข้าข้อมูล เพื่อนำมาวิเคราะห์และทำนายตำแหน่งภายในอาคาร และ 3) ส่วนส่งออกข้อมูล โดยจะส่งผลการทำนายตำแหน่งที่ได้จากส่วนประมวลผลแสดงกลับไปยังโปรแกรมประยุกต์บนโทรศัพท์เคลื่อนที่

ผู้จัดทำได้ศึกษาการทำงานของอุปกรณ์ส่งสัญญาณและเลือกใช้ไมโครคอนโทรลเลอร์ ESP32 เก็บข้อมูลค่า RSSI เพื่อนำค่ามาปรับใช้ในการวิเคราะห์และสร้างแบบจำลองในการระบุตำแหน่งภายในอาคาร โดยจากการศึกษาและพัฒนาผู้จัดทำได้เลือกวิธีระบุตำแหน่งโดยใช้โครงข่ายเซลล์ประสาท ซึ่งหลังจากพัฒนาระบบเพื่อให้ระบุตำแหน่งได้อย่างแม่นยำภายในห้องทดลองสามารถระบุตำแหน่งได้ตามจุดประสงค์ ซึ่งมีระยะความแม่นยำเฉลี่ยคือ 0.4669 เมตร

5.2 ข้อเสนอแนะ

แม้ว่าการทดสอบระบบการระบุตำแหน่งด้วยโครงข่ายเซลล์ประสาทจะให้ผลลัพธ์เป็นที่พึงพอใจ แต่ในแง่ของการนำมาใช้จริงยังมีปัญหาของแม่นยำในช่วงระยะทางหนึ่งของเครื่องส่งสัญญาณความถี่ 2.4 GHz ทำให้ต้องมีการติดตั้งอุปกรณ์ที่มากขึ้นในพื้นที่ขนาดใหญ่ แนวทางในการแก้ปัญหาเหล่านี้คือ ควรใช้อัตราส่วนในการระบุตำแหน่งที่เพิ่มมากขึ้น แลกกับความละเอียดในการระบุตำแหน่งที่ลดลง อีกทั้งยังช่วยลดความซับซ้อนและเวลาในการติดตั้งอีกด้วย และสามารถนำไปประยุกต์ใช้งานกับเครื่องส่งที่มีความถี่ต่ำกว่า และอุปกรณ์มาตรฐานอื่น เพื่อให้ผลลัพธ์ที่ดีขึ้นหรือใช้เทคโนโลยีหรือกระบวนการอื่นเพิ่มเติม เพื่อให้สามารถลดระยะคลาดเคลื่อนในการระบุตำแหน่งให้มีระยะคลาดเคลื่อนน้อยที่สุด

บรรณานุกรม

[1] “เทคโนโลยี Internet of Things และนโยบาย Thailand 4.0.”
http://www.nbtc.go.th/getattachment/Services/quarter2560/ปี-2561/32279/เอกสารแนบ.pdf.aspx.

[2] “รายงานผลการศึกษา เรื่อง ความเป็นไปได้ในการขยายคลื่นความถี่ย่าน 920 - 925 MHz เพื่อรองรับเทคโนโลยีใหม่.”
http://www.nbtc.go.th/getattachment/Spectrum_management/บริการข่าวสาร/รายงานผลการศึกษา/48718/รายงานผลการศึกษา-เรื่อง-ความเป็นไปได้ในการขยายคลื่นความถี่ย่าน-920-925-MHz-เพื่อรองรับเทคโนโลยีใหม่.pdf.aspx.

[3] “Wi-Fi Alliance ประกาศมาตรฐาน Wi-Fi HaLow ใหม่ สำหรับ Internet of Things.”
https://www.techtalkthai.com/wi-fi-alliance-announce-new-wi-fi-standard-called-halow/.

[4] “WI-FI ย่านความถี่ 5G ดีกว่า 2.4 อย่างไร.”
http://www.phuketconnect.co.th/Article/Detail/72319.

[5] “การหาระยะทางในระบบเครือข่ายเซ็นเซอร์ไร้สายโดยใช้ค่า RSSI.”
http://www3.eng.psu.ac.th/pec/6/pec6/paper/CoE/PEC6OR170.pdf.

[6] Botta, Miroslav & Simek, Milan. Adaptive Distance Estimation Based on RSSI in 802.15.4 Network, pp. 1162-1168. 22nd ed. Radioengineering, 2013.

[7] KANG, J., KIM, D., KIM, Y. “RSS self-calibration protocol for WSN localization.” In The 2nd International Symposium on Wireless Pervasive Computing ISWPC. San Juan (Puerto Rico), 2007.

[8] “เจาะลึกเรื่อง AI ทำความรู้จักกับ Machine Learning กับ Deep Learning.”
https://www.pi-tech.biz/17242897/เจาะลึกเรื่อง-ai-ทำความรู้จักกับ-machine-learning-กับ-deep-learning.

[9] บัญชา ปะสีละเตสัง. สร้างการเรียนรู้สำหรับ AI ด้วย Python Machine Learning. กรุงเทพฯ ซีเอ็ดยูเคชั่น, 2564.

[10] Divya Sheel. “Deep learning.”
https://new.abb.com/news/detail/58004/deep-learning.

[11] Jojo John Moolayil. “A Layman’s Guide to Deep Neural Networks.” https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb.

[12] Pallavi Bharadwaj. “Perceptron and its implementation in Python.” https://www.pallavibharadwaj.com/perceptron-and-its-implementation-in-python.

[13] Toyas Dhake. “การตัดสินใจในหุ่นยนต์โดยใช้ Evolutionary Neural Networks.” https://ichi.pro/th/kar-tadsin-ci-ni-hun-ynt-doy-chi-evolutionary-neural-networks-56252249048745.

[14] Pragati Baheti. “12 Types of Neural Network Activation Functions: How to Choose?.” https://www.v7labs.com/blog/neural-networks-activation-functions.

[15] Mr.PL. “Deep Learning แบบฉบับสามัญชน.” https://medium.com/mmp-li/deep-learning แบบฉบับสามัญชน-ep-2-optimization-activation-function เรียนกันสบายๆสไตส์ชิลๆ-9feb5a87e3b2.

[16] Mr.PL. “Gradient Descent.” https://medium.com/mmp-li/gradient-descent-คีย์เวิร์ดที่-data-science-ต้องรู้จักmachine-learning-101-b5c78c1787a0.

[17] Agnes Sauer. “Quick Guide to Gradient Descent.” https://morioh.com/p/15c995420be6.

[18] Keng Surapong. “Loss Function ep.2.” https://www.bualabs.com/archives/1928/what-is-mean-absolute-error-mae-mean-squared-error-mse-root-mean-squared-error-rmse-loss-function-ep-2/.

[19] Keng Surapong. “Loss Function.ep.1.” https://www.bualabs.com/archives/2673/what-is-loss-function-costfunction-error-function-loss-function-how-cost-function-work-machinelearning-ep-1/.

[20] MYITALIANITA. “Overfitting / Underfitting – How Well Does Your Model Fit?.”

https://meditationsonbianddatascience.com/2017/05/11/overfitting-underfitting-how-well-does-your-model-fit/.

[21] Keng Surapong. “BatchNorm คืออะไร.” https://www.bualabs.com/archives/2617/what-is-batchnorm-teachbatch-normalization-train-machine-learning-model-deep-convolutionalneural-network-convnet-ep-5/.

[22] Keng Surapong. “Dropout คืออะไร.” https://www.bualabs.com/archives/1533/what-is-dropout-benefitsdropout-reduce-overfit-deep-learning-training-model-deep-neuralnetwork-regularization-ep-2.

[23] Achieve.Plus. “4 Classification ที่สำคัญใน Supervised Learning.” https://medium.com/achieve-space/4-classification-ที่สำคัญใน-supervised-learning-a64e75250141.

[24] Achieve.Plus. “รู้จักใช้ Scikit-learn เหมือนมีโปรในเกม.” https://medium.com/achieve-space/รู้จักใช้-scikit-learn-เหมือนมีโปรในเกม-4cebd3195518.

[25] Hizoka. “มาทำความรู้จักกับ Flutter กันเถอะ.” https://medium.com/@hizokaz/มาทำความรู้จักกับ-flutter-กันเถอะ-4dca2ad634bd.

[26] Tanapoj Chaivanichanan. “Dart 101: ทำความรู้จักภาษา Dart ฉบับโปรแกรมเมอร์.” https://www.centrilliontech.co.th/blog/2570/dart-101-introduction-for-programmers/.

[27] noomerZx. “Cloud Firestore คุณค่าที่คุณคู่ควร.” https://stories.sellsuki.co.th/cloud-firestore-คุณค่าที่คุณคู่ควร-b471d9b926e3.

[28] Saixiii. “API คืออะไร ใช้ทำอะไร เป็น Applications Program Interface.” https://saixiii.com/what-is-api/.

[29] wannaphong. “FastAPI – Freamwork สร้าง API ด้วย Python ที่ทั้งง่ายและเร็ว.” https://python3.wannaphong.com/2020/02/fastapi-framework-api-python.html.

[30] K. Benkic, M. Malajner, P. Planinsic and Z. Cucej, “Using RSSI value for distance estimation in wireless sensor networks based on ZigBee.” In 2008 15th International Conference on Systems, Signals and Image Processing, pp. 303-306. 2008.

ภาคผนวก ก

คำสั่งส่งสัญญาณของเครื่องส่งสัญญาณ

```
#include<WiFi.h>
#define SSID_AP_NAME "ESP32 AP mode"
#define SSID_AP_PASS NULL
#define SSID_NAME "Ter's Phone2"
#define SSID_PASS "12345678"

void setup() {
        Serial.begin(9600);
        WiFi.mode(WIFI_AP_STA);
        delay(100);
        WiFi.softAP(SSID_AP_NAME, SSID_AP_PASS);
        Serial.print("AP IP address: ");
        Serial.println(WiFi.softAPIP());
        Serial.println();
        Serial.print("Connecting to ");
        Serial.println(SSID_NAME);
        WiFi.begin(SSID_NAME, SSID_PASS);
        while (WiFi.status() != WL_CONNECTED) {
            delay(500);
            Serial.print(".");
        }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("STA IP adress: ");
  Serial.println(WiFi.localIP());
  Serial.println();
}

void loop()
{
}
```

ภาคผนวก ข

คำสั่งที่ใช้ในการประเมินผลการทดสอบ
การรับค่า RSSI และการหาพารามิเตอร์ของแบบจำลอง Path Loss

```
import math
import matplotlib.pyplot as plt
import pandas as pd

def findEta (d, RSSi):
        d0 = 1
        RSSi_d0 = -57
        eta = (RSSi_d0 - RSSi)/(10*(math.log10(d/d0)))
        return eta

def findrssi (d):
        if d == 0: #divided by zero is not define
                d = 0.01
        d0 = 1
        RSSi_d0 = -57
        eta = 2.5690124226916993
        RSSi = RSSi_d0 - 10*eta*(math.log10(d/d0))
        if RSSi > -30: #RSSI less than -30 dBm is not practical
                RSSi = -30
        return RSSi

def find_sd(x, mean):
        all_sum = 0
        for i in range(len(x)):
                sum_sqare = (x[i] - mean)**2
                all_sum += sum_sqare
        sd = math.sqrt((all_sum) / (len(x) - 1))
        return sd

dataframe = pd.read_csv('data.csv')
data = pd.DataFrame.to_dict(dataframe)
```

```
all = {}
rssi = []
d = []
for i in data:
        lst = []
        for j in data[i]:
                lst.append(data[i][j])
        all[int(i)/100] = lst
        rssi.append(math.fsum(lst)/len(lst))
        d.append(int(i)/100)
export = {'d':{}, 'mean':{}, 'mode':{}, 'min':{}, 'max':{}, 'sd':{}}
cnt = 0
for i in all:
        export['d'][cnt] = d[cnt]
        export['mean'][cnt] = rssi[cnt]
        export['mode'][cnt] = max(set(all[i]), key=all[i].count)
        export['min'][cnt] = min(all[i])
        export['max'][cnt] = max(all[i])
        export['sd'][cnt] = find_sd(all[i], rssi[cnt])
        cnt += 1
PDexport = pd.DataFrame(export)
PDexport.to_csv(r'RSSI_test.csv', index=False)

plt.plot(d, rssi)

all_eta = []
for i in range(len(d)):
        if d[i] == 1: #pass for ref at 1 m.
                pass
        else:
                all_eta.append(findEta(d[i], rssi[i]))
```

```
eta = math.fsum(all_eta)/len(all_eta)
print(eta)

rssi2 = []
d2 = []
for i in range(1500):
        j = i/100
        rssi2.append(findrssi(j))
        d2.append(j)
plt.plot(d2, rssi2)
```

ภาคผนวก ค

คำสั่งที่ใช้ในการจำลองชุดข้อมูล

```
import math
import pandas as pd
import matplotlib.pyplot as plt

def findDistance (a, b):          #ฟังก์ชันหาระยะระหว่างจุดกับdevice
        distance = math.sqrt( (a[0]-b[0])**2 + (a[1]-b[1])**2 )
        return distance

def findRSSi (d, ratio):          #ฟังก์ชันหาค่า rssi จากระยะทาง
        d = d*ratio               #แปลงหน่วยให้เป็นเมตรตาม ratio
        if (d < 0.1):             #กรณีระยะน้อยมากให้ fix
                d = 0.1
        d0 = 1                    #ระยะ reference
        RSSi_d0 = -58.889         #RSSI reference
        eta = 1.3561              #path loss exponenent
        RSSi = RSSi_d0 - 10*eta*(math.log10(d/d0))
        return RSSi

def ConvertToDistance (RSSi, ratio):   #ฟังก์ชันหาค่า ระยะทาง จาก rssi
        d0 = 1
        RSSi_d0 = -58.889
        eta = 1.3561
        d = d0*10**((RSSi_d0-RSSi)/10/eta)
        d = d/ratio   #แปลงเมตรเป็นหน่วยตาม ratio
        return d

def listToDict(a_list): #ฟังก์ชันแปลงลิสต์เป็นดิกเชินแนรีส์
        a_dict = {}
        for i in range(len(a_list)):
                a_dict['{}'.format(i)] = a_list[i]
        return a_dict
```

```
#สร้างจุดพิกัดในพื้นที่ ให้อัตราส่วนอยู่ที่ 1 หน่วยพิกัด = 0.3 ม. เมื่อห้องมีขนาด 6x6 เมตร
res_x = 20
res_y = 20
ratio = 0.3                    #กำหนดอัตรส่วนระยะทางต่อหนึ่งหน่วยพิกัด(เมตร)
point = {'position': [], 'x': [], 'y': []}
count = 0
for i in range(res_x+1):
        x = i
        for j in range(res_y+1):
                y = j
                point['position'].append(count)
                point['x'].append(x)
                point['y'].append(y)
                count += 1

#สร้างจุดตำแหน่งdevice 12 ตัว
device = {'x': [], 'y': []}
xx = [0, 0, 0, res_x*1/4, res_x*2/4, res_x*3/4, res_x, res_x, res_x, res_x*1/4, res_x*2/4,
        res_x*3/4]              #ระบุตำแหน่งdeviceในแกน x
yy = [res_y*1/4, res_y*2/4, res_y*3/4, res_y, res_y, res_y, res_y*1/4, res_y*2/4,
        res_y*3/4, 0, 0, 0]     #ระบุตำแหน่งdeviceในแกน y ตามลำดับของพิกัดบนแกน x
distance_data = {}
rssi_data = {}
samp_data = {}
for i in range(len(xx)):
        device['x'].append(xx[i])
        device['y'].append(yy[i])
        distance_data['d_{}'.format(i)] = []
        rssi_data['rssi_{}'.format(i)] = []
        samp_data['rssi_{}'.format(i)] = []
```

```
#สร้างลิสต์ระยะทางจากจุดถึงdevice
for i in range(len(point['x'])):
        p = [point['x'][i], point['y'][i]]
        for j in range(len(device['x'])):
                n_device = [device['x'][j], device['y'][j]]
                d = findDistance(p, n_device)
                distance_data['d_{}'.format(j)].append(d)

#สร้างลิสต์ข้อมูล RSSI จากการแปลงระยะทาง
for i in range(len(distance_data)):
        for j in range(len(distance_data['d_0'])):
                rssi = findRSSi(distance_data['d_{}'.format(i)][j], ratio)
                rssi_data['rssi_{}'.format(i)].append(rssi)

#convert list to dict
rssi_dataset = {}
for i in range(len(device['x'])):
        rssi_dataset['rssi_{}'.format(i)] = listToDict(rssi_data['rssi_{}'.format(i)])
rssi_dataset['position'] = listToDict(point['position'])
rssi_dataset['x'] = listToDict(point['x'])
rssi_dataset['y'] = listToDict(point['y'])
rssi_dataset = pd.DataFrame(rssi_dataset)
rssi_dataset.to_csv(r' RSSI_Dataset.csv', index=False)

#เพิ่ม sample จาก 1 เป็น 50
samp = 50
samp_data['position'] = []
samp_data['x'] = []
samp_data['y'] = []
for i in range(len(rssi_data['rssi_0'])):
        for j in range(samp):
```

```
                samp_data['position'].append(i)
                samp_data['x'].append(point['x'][i])
                samp_data['y'].append(point['y'][i])
                for k in range(len(rssi_data)):
                        s = rssi_data['rssi_{}'.format(k)][i]
                        samp_data['rssi_{}'.format(k)].append(s)

#AWGN
for i in range(len(point['x'])):
        for j in range(len(device['x'])):
                n = np.random.normal(0, 0.5, samp)
                for k in range(samp):
                        new_rssi = samp_data['rssi_{}'.format(j)][samp*i+k] + n[k]
                        samp_data['rssi_{}'.format(j)][samp*i+k] = new_rssi

#convert list to dict
for i in range(len(device['x'])):
        samp_data['rssi_{}'.format(i)] = listToDict(samp_data['rssi_{}'.format(i)])
samp_data['position'] = listToDict(samp_data['position'])
samp_data['x'] = listToDict(samp_data['x'])
samp_data['y'] = listToDict(samp_data['y'])
data = pd.DataFrame(samp_data)
data = data.sample(frac=1)
data.to_csv(r' 0.3_20.csv', index=False)

#plot map
plt.scatter(x = point['x'], y = point['y'], color='blue')
plt.scatter(x = device['x'], y = device['y'], color='red')
plt.show()
```

ภาคผนวก ง

คำสั่งที่ใช้ในการฝึกฝนและประเมินแบบจำลอง

```
import math
import numpy as np
import pandas as pd
from fastai import *
from fastai.tabular import *
from sklearn.metrics import classification_report

#########################################################
###################### 1. Map Dataset ###################
#########################################################
res_x = 7
res_y = 8
point = {'position': [], 'x': [], 'y': []}
count = 0
for i in range(res_y+1):
    y = res_y-i
    for j in range(res_x+1):
        x = j
        point['position'].append(count)
        point['x'].append(x)
        point['y'].append(y)
        count += 1
#print(point)

#########################################################
######################### 2. Train  #####################
#########################################################
def findDistance (a, b): #โครงสร้างเป็น a=[x,y] b=[x,y]
    ratio = 0.4
    distance = ratio*math.sqrt( (a[0]-b[0])**2 + (a[1]-b[1])**2 )
    return distance
```

```
fname = 'co_dataset40'
path = ''
dataframe = pd.read_csv(path+fname+'.csv')
dataframe = dataframe.sample(frac=1)
continuous_column_names = ['rssi_0', 'rssi_1', 'rssi_2', 'rssi_3']
dependent_variable = 'position'
preprocesses = [FillMissing, Categorify, Normalize]

batchsize = 256
nh = [60, 120, 60]
idx_80 = len(dataframe) - int(len(dataframe)*0.2)
valid_idx = list(range(idx_80, len(dataframe)))

tabularlist_test = TabularList.from_df(dataframe.iloc[idx_80:len(dataframe)].copy(),
                                       path=path,
                                       cont_names=continuous_column_names,
                                       procs=preprocesses)
databunch = (TabularList.from_df(dataframe, path=path,
                                 cont_names=continuous_column_names,
                                 procs=preprocesses)
                        .split_by_idx(valid_idx)
                        .label_from_df(cols=dependent_variable)
                        .add_test(tabularlist_test)
                        .databunch(bs=batchsize, num_workers=0))
learner = tabular_learner(databunch, layers=nh,
                          metrics=accuracy,
                          callback_fns=ShowGraph)
learner.fit_one_cycle(600, 0.0002)
learner.save(fname)
learner.load(fname)
```

```
preds = [int(learner.predict(dataframe.iloc[idx])[0]) for idx in valid_idx]
actuals = dataframe.iloc[valid_idx].position.values
report = classification_report(y_pred=preds, y_true=actuals, output_dict=True)

case = []
for i in range(len(actuals)):
  f = (actuals[i], preds[i])
  case.append(f)

a = []
b = []
for i in range(len(case)):
  co_actuals = ((point['x'][case[i][0]]), (point['y'][case[i][0]]))
  a.append(co_actuals)
  co_preds = ((point['x'][case[i][1]]), (point['y'][case[i][1]]))
  b.append(co_preds)

all_dist = []
for i in range(len(a)):
  dist = findDistance(a[i], b[i])
  all_dist.append(dist)

average = np.mean(all_dist)
sd = np.std(all_dist)
min = np.min(all_dist)
max = np.max(all_dist)

print('accuracy = %.4f' %(report['accuracy']*100), 'mean = %.4f' %(average),
                'min = %.4f' %(min), 'max = %.4f' %(max), 'sd = %.4f' %(sd))
```

```
subpath = ''
report = pd.DataFrame(report)
report = report.T
report.to_csv(path+subpath+'report_'+fname+'.csv', index=True)
map = pd.DataFrame(point)
map.to_csv(path+subpath+'map_'+fname+'.csv', index=False)
```

ภาคผนวก จ

คำสั่งของเซิร์ฟเวอร์ประมวลผล และคำสั่งที่ใช้ในการสร้าง API

```
from fastapi import FastAPI
from pydantic import BaseModel
from typing import Optional
from fastapi.middleware.cors import CORSMiddleware
import uvicorn
import pandas as pd
from fastai import *
from fastai.tabular import *
from sklearn.metrics import classification_report

app = FastAPI()
origins = [
    "*"
]
app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],)

def findPosition (rssi):
    fname = 'co_dataset40'
    path = ''
    dataframe = pd.read_csv(path+fname+'.csv')
    dependent_variable = 'position'
    continuous_column_names = ['rssi_0', 'rssi_1', 'rssi_2', 'rssi_3']
    preprocesses = [FillMissing, Normalize]
    idx_80 = len(dataframe) - int(len(dataframe)*0.2)
    valid_idx = list(range(idx_80, len(dataframe)))
    batchsize = 256
```

```
    nh = [60, 120, 60]
    tabularlist_test = TabularList.from_df(dataframe.iloc[idx_80:len(dataframe)].copy(),
                                path=path,
                                cont_names=continuous_column_names,
                                procs=preprocesses)
    databunch = (TabularList.from_df(dataframe, path=path,
                            cont_names=continuous_column_names,
                            procs=preprocesses)
                        .split_by_idx(valid_idx)
                        .label_from_df(cols=dependent_variable)
                        .add_test(tabularlist_test)
                        .databunch(bs=batchsize, num_workers=0))
    learner = tabular_learner(databunch, layers=nh,
                            metrics=accuracy,
                            callback_fns=ShowGraph)
    learner.load(fname)
    return learner.predict(rssi)

class RSSI(BaseModel):
    rssi_0: Optional[float] = None
    rssi_1: Optional[float] = None
    rssi_2: Optional[float] = None
    rssi_3: Optional[float] = None
    rssi_4: Optional[float] = None
    rssi_5: Optional[float] = None
    rssi_6: Optional[float] = None
    rssi_7: Optional[float] = None
    rssi_8: Optional[float] = None
    rssi_9: Optional[float] = None
    rssi_10: Optional[float] = None
    rssi_11: Optional[float] = None
```

```
class MyValue:
    def __init__(self, rssi, pos):
        self.rssi = rssi
        self.pos = pos


@app.get("/rssi/")
async def getrssi():
    return MyValue.rssi

@app.get("/pos/")
async def getpos():
    return MyValue.pos

@app.post("/rssi/")
async def postrssi(input: RSSI):
    MyValue.rssi = input.dict()
    rssi = input.dict()
    loc = int(findPosition(rssi)[0])
    path = ''
    fname = 'co_dataset40'
    map = pd.read_csv(path+'map_'+fname+'.csv')
    x = map['x'][loc]
    y = map['y'][loc]
    MyValue.pos = {"x": int(x), "y": int(y)}
    return {"x": int(x), "y": int(y)}

if __name__ == '__main__':
                    uvicorn.run(app, host='http://161.246.18.222', port=8888)
```