โปรแกรมแปลภาษามืออัตโนมัติ

Hand Sign Detection for Thai Sign Language

เจตณัฐ มหาศักดิ์ศิริ

Jetanat Mahasaksiri

ณัฐดนัย หายทุกข์

Nutdanai Haitook

อนัญลักษณ์ มหาสุวรรณ์

Ananyalak Mahasuwan

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2565

โปรแกรมแปลภาษามืออัตโนมัติ

Hand Sign Detection for Thai Sign Language

โดย

เจตณัฐ มหาศักดิ์ศิริ

ณัฐดนัย หายทุกข์

อนัญลักษณ์ มหาสุวรรณ์

อาจารย์ที่ปรึกษา

ผศ.ดร. ยุทธนา คิดใจเดียว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2565

ปริญญานิพนธ์ปีการศึกษา 2565

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมแปลภาษามืออัตโนมัติ

Hand Sign Detection for Thai Sign Language

ผู้จัดทำ นายเจตณัฐ มหาศักดิ์ศิริ รหัสนักศึกษา 62010138

นายณัฐดนัย หายทุกข์ รหัสนักศึกษา 62010269

นายอนัญลักษณ์ มหาสุวรรณ์ รหัสนักศึกษา 62011007

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ผศ.ดร. ยุทธนา คิดใจเดียว

อาจารย์ที่ปรึกษา

หัวข้อโครงงาน	โปรแกรมแปลภาษามืออัตโนมัติ	
นักศึกษา	นายเจตณัฐ มหาศักดิ์ศิริ	รหัสนักศึกษา 62010138
	นายณัฐดนัย หายทุกข์	รหัสนักศึกษา 62010269
	นายอนัญลักษณ์ มหาสุวรรณ์	รหัสนักศึกษา 62011007
ปริญญา	วิศวกรรมศาสตร์บัณฑิต	
ภาควิชา	วิศวกรรมอิเล็กทรอนิกส์	
ปีการศึกษา	2565	
อาจารย์ที่ปรึกษาโครงการ	ผศ.ดร. ยุทธนา คิดใจเดียว	

บทคัดย่อ

ปัจจุบันเทคโนโลยี AI กำลังพัฒนาและถูกใช้งานกันอย่างแพร่หลาย พวกเราสนใจและได้เล็งเห็นถึงความสามารถของเทคโนโลยีนี้ จึงนำมาพัฒนาเป็นโปรแกรมแปลภาษามือสำหรับผู้พิการทางการพูดและการได้ยินเพราะอยากช่วยให้พวกเขาได้ใช้ชีวิตอย่างสะดวกมากขึ้นและให้คนทั่วไปสามารถเข้าใจภาษามือได้ โดยโครงงานนี้ได้นำเสนอโปรแกรมแปลภาษามือที่ใช้เฟรมเวิร์คของ Google ที่มีชื่อว่า MediaPipe ในการเก็บคุณลักษณะจุดต่างๆบนมือและบนร่างกายร่วมกับ LSTM Model (Long Short Term Memory Model) ซึ่งเป็น Deep Learning Model สำหรับการเรียนรู้ข้อมูลและใช้ Multi-head attetion ในกระบวนการของ Transformer แล้วนำค่าดัชนีต่าง ๆที่บ่งบอกถึงความแม่นยำของแต่ละท่ามาเปรียบเทียบกันและแสดงผลการทำนายท่าท่างที่ได้รับแบบเรียลไทม์ ซึ่งการใช้ LSTM Model ร่วมกับ Multi-head attetion ได้ค่าความแม่นยำที่ 98.28% โดยท่าที่ใช้ทดสอบมีจำนวน 51 ท่า ได้แก่ สบายดี, เด็ก, สวัสดี, ขอโทษ, ขยั่น, เมื่อวาน, เข้าใจ, พวกเรา, เศร้า, ขอบคุณ, ง่าย, พวกเขา, ชื่อ, ที่ไหน, รัก, ต้องการ, ทำ, ผู้ชาย, อิ่ม, หัวเราะ, กิน, ชอบ, มา, ไม่เข้าใจ, ร้องไห้, งาน, ทำไม, คิดถึง, แม่, ยิ้ม, พรุ่งนี้, ฉัน, อะไร, หิว, สงสัย, โกรธ, กลัว, คุณ, ขอบคุณมาก, พ่อ, ขี้เกียจ, ไป, เมื่อไร, ผู้หญิง, ไม่ชอบ, ลืม, มีความสุข, วันนี้, ยาก, อย่างไรและจำได้

Project Title	Hand Sign Detection for Thai Sign Language	
Student	Mr.Jetanat Mahasaksiri	Student ID 62010138
	Mr.Nutdanai Haitook	Student ID 62010269
	Mr.Ananyalak Mahasuwan	Student ID 62011007
Degree	Bachelor of Engineering	
Program	Electronics Engineering	
Year	2022	
Project Advisor	Asst. Prof. Yuttana Kitjaidure, Ph.D	

ABSTRACT

Currently, AI technology is developing and being widely used. We are interested and have seen the capabilities of this technology. Therefore, it was developed into a sign language translator program for people with speech and hearing disabilities because they want to help them live more conveniently and allow people to understand sign language. This project presents a sign language translator program that uses Google's framework called MediaPipe to store hand and body features together with LSTM Model (Long Short Term Memory Model), a Deep Learning Model for Learn data and use Multi-head attetion in Transformer process and take different index values. That indicates the accuracy of each posture to compare and show the results of posture prediction obtained in real time. Using the LSTM model in conjunction with Multi-head attetion, the accuracy was 98.28%. There are 51 poses for the test, namely, well, baby, hello, sorry, shaken, yesterday, understand, us, sad, thank you, easy, they, name, where, love, want, do, man, full, laugh, eat, like, come, don't understand, cry, work, why, miss, mom, smile, tomorrow, me, what, hungry, wonder, angry, scared, you, thank you very much, dad, lazy, go, when? , woman, dislike, forget, happy, today, difficult, how and remember

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้จากความรู้ที่ผู้จัดทำได้รับจาก ผศ.ดร. ยุทธนา คิดใจเดียว ที่ได้ให้คำแนะนำเรื่องที่วิธีการสร้างโมเดลและวิธีกาคิดต่างๆที่เหมาะกับปริญญานิพนธ์ ให้คำปรึกษากับแนวทางในการทำปริญญานิพนธ์นี้ให้สมบูรณ์ และคำแนะนำจากเพื่อนๆ ผู้จัดทำขอขอบคุณทุกๆคนที่ช่วยเหลือให้ปริญญานิพนธ์นี้ประสบความสำเร็จออกมาได้ตามที่ต้องการ

เจตณัฐ มหาศักดิ์ศิริ

ณัฐดนัย หายทุกข์

อนัญลักษณ์ มหาสุวรรณ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ภาษามือเป็นภาษาสำหรับคนหูหนวกที่ใช้ในการติดต่อสื่อสารกัน โดยอาศัยการแสดงออกทางสีหน้า ท่าทาง และการเคลื่อนไหวร่างกายเพื่อประกอบการสื่อสาร การใช้ภาษามือนั้นไม่ใช่ภาษาสากลที่สามารถใช้ได้ครอบคลุมทุกประเทศ เนื่องจากการใช้ภาษามือจะแตกต่างกันออกไปตามลักษณะของภาษาในประเทศหรือท้องถิ่นนั้น ๆ ซึ่งโครงงานนี้ได้ให้ความสนใจกับภาษามือของประเทศไทยโดยเน้นไปที่ภาษามือที่ใช้ในการแสดงออกถึงอารมณ์ และเพื่อให้การสื่อสารของคนหูหนวกเป็นไปอย่างสะดวกและคนทั่วไปสามารถเข้าใจได้ง่าย จึงเป็นที่มาให้เราใช้กระบวนการเรียนรู้ของปัญญาประดิษฐ์เข้ามาพัฒนาในโครงงานชิ้นนี้

1.2 วัตถุประสงค์

1. ฝึกทักษะการเขียนโปรแกรมในภาษา python
2. ศึกษาประยุกต์ใช้งาน machine learning และ deep learning
3. ศึกษาการจัดการและเตรียมข้อมูลโดยเฉพาะข้อมูลที่เป็นรูปแบบของวิดีโอก่อนที่จะนำเข้าสู่กระบวนการ training และ testing

1.3 ขอบเขตการศึกษา

1. สามารถแสดงความหมายของภาษามือได้ 51 ท่า
2. ความแม่นยำของโมเดลมากกว่าร้อยละ 97

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. เพิ่มทักษะในการเขียนโปรแกรมด้วยภาษา python
2. เข้าใจหลักการและการประยุกต์ใช้งานของ machine learning และ deep learning
3. เพิ่มทักษะในการจัดการและเตรียมข้อมูลซึ่งเป็นทักษะที่สำคัญในเรื่องนี้
4. โปรเจคสามารถใช้งานได้จริงเพื่อเป็นประโยชน์แก่ผู้ที่ต้องการสื่อสารกันระหว่างผู้ใช้ภาษามือกับผู้ใช้ภาษาพูด

1.5 ระยะเวลาในการทำโครงงาน

ตั้งแต่วันที่ 31 มกราคม 2566 ถึงวันที่ 25 เมษายน 2566

ตารางที่ 1.1 ระยะเวลาในการทำโครงงาน

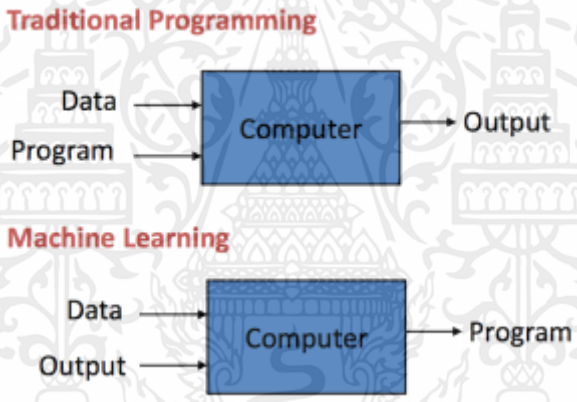
ขั้นตอนการดำเนินงาน	สัปดาห์ที่												
	1	2	3	4	5	6	7	8	9	10	11	12	13
1. ศึกษาทฤษฎี	◄	—	—	—	—	—	►	สอบกลางภาค	◄	—	—	—	►
2. รวบรวมและเรียบเรียงข้อมูล												◄	►
3. สรุปผล													◄►
4. เขียนรายงานฉบับสมบูรณ์													◄►

*** ใช้ ◄——► ในสัปดาห์ที่ดำเนินการในแต่ละขั้นตอน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 Machine Learning

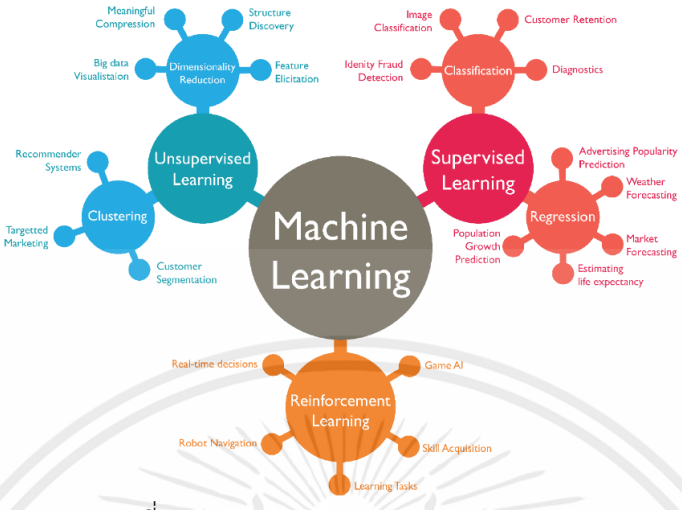
Machine Learning คือ การเรียนรู้ของระบบคอมพิวเตอร์ด้วยตัวเองโดยใช้ข้อมูล ซึ่งแตกต่างกับการเขียนโปรแกรมทั่วไปเพราะ Programming เราจะใส่ข้อมูล (Data) และ Program เข้าไปเพื่อให้ได้ Output แต่ Machine Learning ไม่ได้ Program ไปตรงๆแต่ใช้การใส่ข้อมูลและผลลัพธ์ (Output) เข้าไปเพื่อหารูปแบบที่จะนำไปตอบในอนาคตได้ว่า Input แบบนี้ Output จะเป็นอะไร



รูปที่ 2.1 เปรียบเทียบการเขียนโปรแกรมแบบปกติกับ Machine Learning

แหล่งที่มา: Machine Learning คืออะไร?. เคยสงสัยกันบ้างรึเปล่า ว่า | by Vithan Minaphinant | investic | Medium

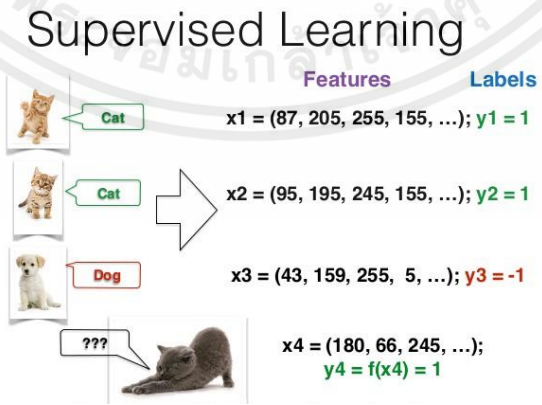
2.1.1 รูปแบบของ Machine Learning



รูปที่ 2.2 ประเภทของ Machine Learning

แหล่งที่มา : Machine Learning คืออะไร?. เคยสงสัยกันบ้างรึเปล่า ว่า | by Vithan Minaphinant | investic | Medium

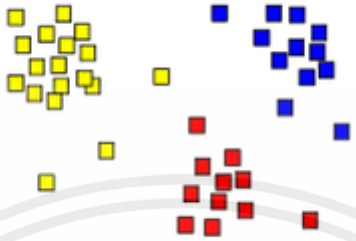
1). Supervised Learning คือ กระบวนการเรียนรู้โดยเรียนรู้จาก Data และ Output ที่กำหนดให้โดยระบุ (Label) ไว้แล้วว่า Input เข้ามาแบบนี้แล้ว Output เป็นแบบไหนแล้วนำไปเข้า Model ที่ทำให้คอมพิวเตอร์สามารถเรียนรู้ จำแนกประเภทของข้อมูลได้ ซึ่งข้อมูลเหล่านั้นจะถูกแปลงให้เป็นภาษาคอมพิวเตอร์ก่อน เช่น การแยกประเภทหมาแมว จะต้องมีลักษณะ (Features) ที่ใช้ในการระบุว่าเป็นหมาหรือแมว เช่น สีตา สีขน ขนาดตัว โดยที่เราได้ระบุผลลัพธ์ไว้แล้วว่าลักษณะแบบไหนเป็นหมาแล้วลักษณะไหนเป็นแมว



รูปที่ 2.3 การแยกประเภทหมาและแมวแบบ Supervised Learning

แหล่งที่มา : Introduction to Deep Learning (Dmytro Fishman Technology Stream) (slideshare.net)

2). Unsupervised Learning คือ กระบวนการการเรียนรู้ที่ไม่มีการระบุ (Label) Data ว่าถ้า Data เข้ามาแบบนี้แล้ว Output ควรเป็นแบบไหน แต่คอมพิวเตอร์จะทำการเรียนรู้และจำแนกจากลักษณะของ Data เพื่อหารูปแบบของ Output เอง



รูปที่ 2.4 Cluster Analysis

แหล่งที่มา : Cluster analysis - Wikipedia

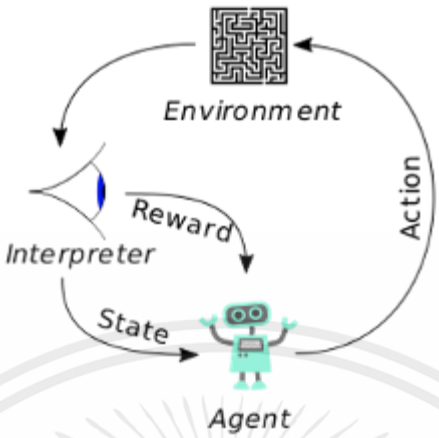
3). Reinforcement Learning คือ กระบวนการการเรียนรู้ที่มีหลักการทำงานเสมือนกับการที่มนุษย์เรียนรู้บางสิ่งบางสิ่งบางอย่างด้วยการลองผิดลองถูก และมีการเรียนรู้เกิดขึ้นระหว่างทางว่าการกระทำไหนดีหรือไม่ดี ซึ่ง Reinforcement Learning ประกอบด้วยองค์ประกอบหลัก ดังต่อไปนี้

- Agent – ผู้กระทำ Action
- Action (a) – การกระทำของ Agent ที่ส่งผลบางอย่างต่อ Environment
- Environment (e) – ระบบที่ Agent ต้องมีปฏิสัมพันธ์ด้วย
- State (s) – สถานการณ์ของ Environment ที่ทาง Agent สามารถรับรู้ได้
- Policy (π) – หลักการที่ Agent ใช้ในการตัดสินใจเลือก Action หลังจากประเมินสถานการณ์แล้ว
- Reward (R) – ตัวประเมินผลลัพธ์ที่เกิดจากการกระทำของ Agent เช่น คะแนน กำไรที่ได้รับ หรือ ผลแพ้ชนะ เป็นต้น

Agent จะรับรู้สถานการณ์ หรือ State ของ Environment จากนั้นจึงใช้ Policy ในการเลือกทำการกระทำบางอย่าง (Action) ที่ส่งผลต่อ Environment ซึ่งผลจากการกระทำ หรือ Action นั้นจะทำให้สถานการณ์ของ Environment เปลี่ยนจากสถานการณ์เดิมไปสู่อีกสถานการณ์หนึ่ง (State ใหม่)

ดังนั้นหลักการของ Reinforcement Learning คือการเรียนรู้ของ Agent ที่เกิดจากปฏิสัมพันธ์แบบลองผิดลองถูกระหว่าง Agent กับ Environment โดย Agent จะสามารถรับรู้สถานการณ์ของ Environment

ผ่าน State และเลือกการกระทำ Action ที่ส่งผลต่อ Environment โดยหวังว่าจะได้ผลลัพธ์ Reward ที่ดีที่สุด รวมทั้งเรียนรู้ผ่านข้อผิดพลาดในอดีตที่เกิดขึ้น



รูปที่ 2.5 ภาพแสดงหลักการทำงานของ Reinforcement Learning

แหล่งที่มา : Reinforcement learning - Wikipedia

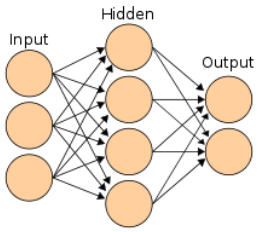
2.2 การเรียนรู้เชิงลึก (Deep Learning)

การเรียนรู้เชิงลึกเป็นส่วนหนึ่งของวิธีการการเรียนรู้ของเครื่องบนพื้นฐานของโครงข่ายปราสาทเทียมและการเรียนเชิงคุณลักษณะ การเรียนรู้สามารถเป็นได้ทั้งแบบการเรียนรู้แบบมีผู้สอน การเรียนรู้แบบกึ่งมีผู้สอน และการเรียนรู้แบบไม่มีผู้สอนคำว่า "ลึก" ในความหมายมาจากการที่มีชั้นของโครงข่ายหลายชั้น ที่มีประสิทธิภาพมากขึ้น การเรียนที่สะดวกขึ้น และการเข้าใจในโครงสร้างที่ชัดเจนขึ้น

พื้นฐานของการเรียนรู้เชิงลึกคือ อัลกอริทึมที่พยายามจะสร้างแบบจำลองเพื่อแทนความหมายของข้อมูลในระดับสูงโดยการสร้างสถาปัตยกรรมข้อมูลขึ้นมาที่ประกอบไปด้วยโครงสร้างย่อย ๆ หลายอัน และแต่ละอันนั้นได้มาจากการแปลงที่ไม่เป็นเชิงเส้น การเรียนรู้เชิงลึกอาจมองได้ว่าเป็นวิธีการหนึ่งของการเรียนรู้ของเครื่องที่พยายามเรียนรู้วิธีการแทนข้อมูลอย่างมีประสิทธิภาพ การเรียนรู้เชิงลึกถือว่าเป็นวิธีการที่มีศักยภาพสูงในการจัดการกับฟีเจอร์สำหรับการเรียนรู้แบบไม่มีผู้สอนหรือการเรียนรู้แบบกึ่งมีผู้สอน ซึ่งการเรียนรู้แบบไม่มีผู้สอนเป็นเทคนิคหนึ่งของการเรียนรู้ของเครื่องโดยการสร้างโมเดลที่เหมาะสมกับข้อมูล การเรียนรู้แบบนี้แตกต่างจากการเรียนรู้แบบมีผู้สอน คือ จะไม่มีการระบุผลที่ต้องการหรือประเภทไว้ก่อน การเรียนรู้แบบนี้จะพิจารณาวัตถุเป็นเซตของตัวแปรสุ่ม แล้วจึงสร้างโมเดลความหนาแน่นร่วมของชุดข้อมูล และการเรียนรู้แบบมีผู้สอน เป็นรูปแบบการเรียนรูปแบบหนึ่งของการเรียนรู้ของเครื่องที่จับคู่ระหว่างข้อมูลนำเข้าและข้อมูลส่งออกตามพื้นฐานตัวอย่างการทำงานอ้างอิงจากข้อมูลสอน ซึ่งประกอบด้วยชุดข้อมูลตัวอย่าง

กระบวนการที่การเรียนรู้เชิงลึกนำไปใช้ได้แก่ การเข้ารหัสประสาท (Neural encoding) อันเป็นกระบวนการหาความสัมพันธ์ระหว่างตัวกระตุ้นกับการตอบสนองของเซลล์ประสาทในสมอง และในการเรียนรู้ของเครื่อง (Machine learning) มีสถาปัตยกรรมการเรียนรู้หลายแบบบนหลักการของการเรียนรู้เชิงลึกนี้ได้แก่

2.2.1 โครงข่ายประสาทเทียมแบบลึก (Deep Artificial Neural Networks)



รูปที่ 2.6 ข่ายประสาทเทียมที่มีการเชื่อมต่อกันผ่านกลุ่มโหนด
แหล่งที่มา : โครงข่ายประสาทเทียม - วิกิพีเดีย (wikipedia.org)

โครงข่ายประสาทเทียม (Artificial neural networks: ANN) หรือ ข่ายงานประสาทเทียม (Connectionist systems) คือระบบคอมพิวเตอร์จากโมเดลทางคณิตศาสตร์ เพื่อจำลองการทำงานโครงข่ายประสาทชีวภาพที่อยู่ในสมองของสัตว์ โครงข่ายประสาทเทียมสามารถเรียนรู้ที่จะทำงานที่มอบหมายได้ จากการเรียนรู้ผ่านตัวอย่าง โดยไม่ถูกโปรแกรมด้วยกฎเกณฑ์ตายตัวแบบระบบอัตโนมัติ โปรแกรมโครงข่ายประสาทเทียมสามารถแยกแยะรูปภาพได้โดยปราศจากการความรู้ก่อนหน้า ว่าสิ่งนั้นคืออะไร (อาทิ แมวมีขน มีหูแหลม มีเขี้ยว มีหาง) แทนที่จะใช้ความรู้ดังกล่าว โครงข่ายประสาทเทียมทำการระบุสิ่งนั้นโดยอัตโนมัติด้วยการระบุลักษณะเฉพาะ จากชุดตัวอย่างที่เคยได้ประมวลผล แนวคิดเริ่มต้นของเทคนิคนี้ได้มาจากการศึกษาโครงข่ายไฟฟ้าชีวภาพ (bioelectric network) ในสมองซึ่งประกอบด้วย เซลล์ประสาท (neurons) และ จุดประสานประสาท (synapses) ตามโมเดลนี้ ข่ายงานประสาทเกิดจากการเชื่อมต่อระหว่างเซลล์ประสาท จนเป็นเครือข่ายที่ทำงานร่วมกัน การประมวลผลต่าง ๆ ของโครงข่ายประสาทเทียมเกิดขึ้นในหน่วยประมวลผลย่อย เรียกว่า โหนด (node) ซึ่งโหนดเป็นการจำลองลักษณะการทำงานมาจากเซลล์การส่งสัญญาณ ในสมองมีความสามารถในการส่งสัญญาณไปยังเซลล์ประสาทเซลล์อื่น ๆ ที่เชื่อมต่อกับมันได้

ในการสร้างระบบโครงข่ายประสาทเทียม เอาต์พุตของแต่ละเซลล์ประสาทจะมาจากการคำนวณผลรวมของอินพุต ด้วยฟังก์ชันการแปลง (transfer function) ซึ่งทำหน้าที่รวมค่าเชิงตัวเลขจากเอาต์พุตของเซลล์ประสาทเทียม แล้วทำการตัดสินใจว่าจะส่งสัญญาณเอาต์พุตออกไปในรูปใด ฟังก์ชันการแปลงอาจเป็นฟังก์ชันเส้นตรงหรือไม่ก็ได้ โครงข่ายประสาทเทียม ประกอบไปด้วย จุดเชื่อมต่อ (Connections) ซึ่งสามารถเรียกสั้น ๆ ได้ว่า เอจ (Edge), เมื่อโครงข่ายประสาทมีการเรียนรู้ จะเกิดค่าน้ำหนักขึ้น, ค่าน้ำหนัก (weights) คือ สิ่งที่ได้จากการเรียนรู้ของโครงข่ายประสาทเทียม หรือเรียกอีกอย่างหนึ่งว่า ค่าความรู้ (knowledge) ค่านี้จะถูกเก็บเป็นทักษะเพื่อใช้ในการจดจำข้อมูลอื่น ๆ ที่อยู่ในรูปแบบเดียวกัน

โครงสร้างของโมเดลข่ายงานประสาทแบบป้อนไปหน้า (feedforward) ประกอบด้วยเซตของบัพ (node) ซึ่งอาจจะถูกกำหนดให้เป็นบัพอินพุต (input nodes) บัพเอาต์พุต (output nodes) หรือ บัพอยู่ระหว่างกลางซึ่งเรียกว่า บัพฮิดเดน (hidden nodes) มีการเชื่อมต่อระหว่างบัพ (หรือนิวรอน) โดยกำหนดค่าน้ำหนัก (weight) กำกับอยู่ที่เส้นเชื่อมทุกเส้น เมื่อข่ายงานเริ่มทำงานจะมีการกำหนดค่าให้แก่บัพอินพุต โดย

ค่าเหล่านี้อาจจะได้มาจากการกำหนดโดยมนุษย์จากเซนเซอร์ที่วัดค่าต่างๆ หรือผลจากโปรแกรมอื่นๆจากนั้น บัพอินพุตจะส่งค่าที่ได้รับไปตามเส้นเชื่อมขาออก โดยที่ค่าที่ส่งออกไปจะถูกคูณกับค่าน้ำหนักของเส้นเชื่อม บัพในชั้นถัดไปจะรับค่า ซึ่งเป็นผลรวมจากบัพต่างๆแล้วจึงคำนวณผลอย่างง่าย โดยทั่วไปจะใช้ฟังก์ชันซิกมอยด์ (sigmoid function) แล้วส่งค่าไปยังชั้นถัดไป การคำนวณเช่นนี้จะเกิดขึ้นไปเรื่อยๆทีละชั้นจนถึงบัพเอาต์พุต

2.2.2 โครงข่ายประสาทเทียมแบบสังวัตนาการ (Convolutional Neural Networks)

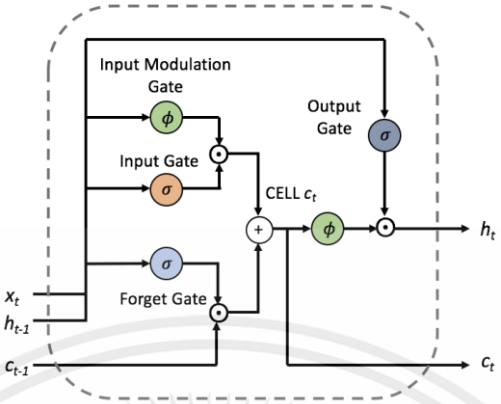
เป็นโครงข่ายประสาทเทียมหนึ่งในกลุ่ม bio-inspired โดยที่ CNN จะจำลองการมองเห็นของมนุษย์ที่มองพื้นที่เป็นที่ย่อย ๆ และนำกลุ่มของพื้นที่ย่อย ๆมาผสานกันเพื่อดูว่าสิ่งที่เห็นอยู่เป็นอะไรกันแน่ การมองพื้นที่ย่อยของมนุษย์จะมีการแยกคุณลักษณะ (feature) ของพื้นที่ แนวคิดของ CNN มีระบบการคำนวณที่สอดคล้องกับ Concept ของมันเองและต้องมีคณิตศาสตร์มารองรับ โดยการคำนวณตามแนวคิดนี้ใช้หลักการเดียวกันกับ คอนโวลูชันเชิงพื้นที่ (Spatial Convolution) ในการทำงานด้าน Image Processing

การคำนวณนี้จะเริ่มจากการกำหนดค่าใน ตัวกรอง (filter) หรือ เคอร์เนล (kernel) ที่ช่วยดึงคุณลักษณะที่ใช้ในการรู้จำวัตถุออก โดยปกติตัวกรอง/เคอร์เนลอันหนึ่งจะดึงคุณลักษณะที่สนใจออกมาได้หนึ่งอย่าง เราจึงจำเป็นต้องตัวกรองหลายตัวกรองด้วย เพื่อหาคุณลักษณะทางพื้นที่หลายอย่างประกอบกันย่อยนั้น เช่น ลายเส้น และการตัดกันของสี ซึ่งการที่มนุษย์รู้ว่าพื้นที่ตรงนี้เป็นเส้นตรงหรือสีตัดกัน เพราะมนุษย์ดูทั้งจุดที่สนใจและบริเวณรอบ ๆประกอบกัน

2.2.3 โครงข่ายประสาทเทียมแบบวนซ้ำ (Recurrent Neural Network)

Recurrent Neural Network (RNN) คือ Artificial Neural Network แบบหนึ่งที่ออกแบบมาแก้ปัญหาสำหรับงานที่ข้อมูลมีลำดับ Sequence โดยใช้หลักการ Feed สถานะภายในของโมเดล กลับมาเป็น Input ใหม่ คู่กับ Input ปกติ เรียกว่า Hidden State, Internal State, Memory ช่วยให้โมเดลรู้จำ Pattern ของลำดับ Input Sequence ได้ RNN มีการพัฒนาต่อยอดไปอีกหลาย Variation ที่เป็นที่นิยมได้แก่ Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU) เพื่อแก้ปัญหาของ RNN ที่มีต่อ sequence ยาวๆ ของข้อมูล ก็เลยมีการเสนอการใช้ Long Short-Term Memory หรือ LSTM นี้ขึ้นมา เป็นเทคนิคหนึ่งที่ถูกพัฒนาจาก Recurrent neural network (RNN) ซึ่ง RNN นั้นมีหลักการทำงาน คือ การนำ Output ที่ได้จากการคำนวณจากโหนดก่อนหน้านี้กลับมาใช้เป็นข้อมูล Input ของโหนดถัดไป ซึ่งแต่ละโหนดของ RNN นั้นจะมีข้อมูลที่เข้ามา 2 ส่วน คือ ข้อมูล Input ของโหนดนั้นๆกับ Output ที่ผ่านการคำนวณจากโหนดก่อนหน้า โดยข้อมูลทั้ง 2 ชุดที่เข้ามาในโหนดจะถูกรวมเข้าด้วยกัน ก่อนจะถูกแยกผลลัพธ์ออกเป็น 2 ส่วน คือ ผลลัพธ์ที่ได้จากโหนดนั้น ๆ และผลลัพธ์ที่จะถูกนำไปเป็นข้อมูล Input ของโหนดถัดไป เทคนิค RNN นั้นเหมาะนำมาใช้งานกับข้อมูลที่มีลักษณะเป็นลำดับ (Sequence) หรือข้อมูลที่มีความต่อเนื่อง เช่น ข้อมูลอนุกรมเวลา (Time Series), ข้อมูลเสียง, ข้อมูลประเภทข้อความ, ข้อมูลภาพและวิดิทัศน์ เป็นต้น

2.2.4 LSTM (Long Short Term Memory)



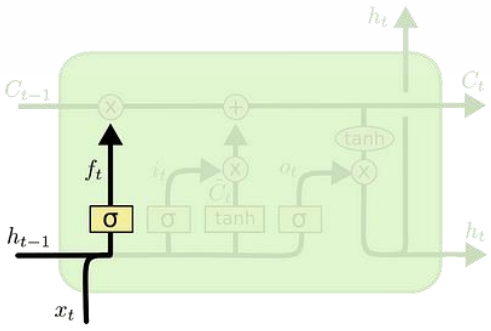
รูปที่ 2.7 การทำงานของ LSTM

แหล่งที่มา : Long Short-Term Memory (LSTM). คิดว่าหลายๆ คนที่เคยทำ machine learning... | by Sirinart Tangruamsub | Medium

Long short-term memory (LSTM) เป็นโครงข่ายประเภท RNN รูปแบบหนึ่งที่ถูกพัฒนาขึ้นมาให้มีความเสถียรและมีประสิทธิภาพมากขึ้น โดยมีหลักการทำงานคือ สามารถเก็บ ‘สถานะ’ หรือข้อมูลของแต่ละโหนดเอาไว้เพื่อที่เวลาย้อนกลับไปดูจะได้ทราบถึงที่มาของข้อมูลค่าดังกล่าวว่าเดิมเป็นค่าอะไร และจุดเด่นของแบบจำลอง LSTM คือฟังก์ชันพิเศษที่มีหน้าที่เสมือนประตู(Gate) ที่คอยความคุมข้อมูลที่จะเข้าในแต่ละโหนด ซึ่งประกอบด้วย Forget gate layer, Input gate layer และ Output gate layer

ในส่วน Forget gate layer ในโครงสร้างของ LSTM จะมี Sigmoid Layer ซึ่งให้ค่าออกมาระหว่าง 0 กับ 1 จะได้ค่าออกมา ซึ่งจะนำไปใช้ในการคูนกับ State ก่อนในภายหลังตามสมการที่ 2.1 เป็นการปรับการใช้ State เก่าโดยจะเลือกว่าจะจำข้อมูลจาก state ก่อนหน้าหรือไม่

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \qquad (2.1)$$



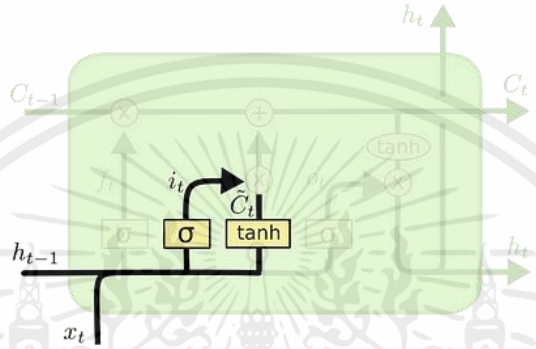
รูปที่ 2.8 องค์ประกอบของ Forget gate layer

แหล่งที่มา : Sirawich Smitsomboon – Medium

ส่วนถัดมาเป็น Input gate layer จะมีการคำนวน (ในส่วนของ tanh) ตามสมการที่ 2.3 แล้วนำค่านั้นไปคูณกับค่าที่ได้จาก Sigmoid Layer ตามสมการที่ 2.2 เพื่อตั้งค่า Weight ในข้อมูลใหม่และเพิ่มเข้าไปใน cell state

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.2)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.3)$$

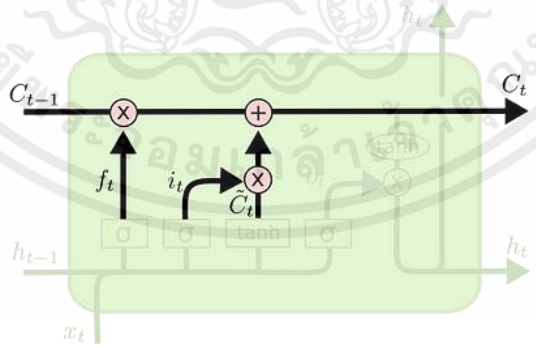


รูปที่ 2.9 องค์ประกอบของ Input gate layer

แหล่งที่มา : Sirawich Smitsomboon – Medium

จากนั้นนำสองส่วนมารวมกันโดยลืมค่าเก่าบางส่วนและรับบางส่วนจากของใหม่ จะได้ค่า Cell State ตามสมการที่ 2.4

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (2.4)$$



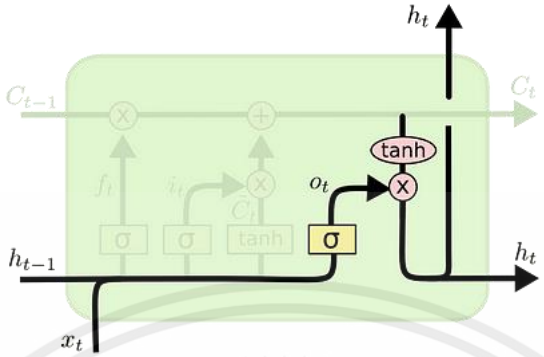
รูปที่ 2.10 การรับค่าเข้า cell state

แหล่งที่มา : Sirawich Smitsomboon – Medium

ส่วนสุดท้ายทำการนำค่า Cell State มาคำนวน (tanh) ตามสมการที่ 2.6 และนำค่าที่ได้มาคูณกับค่าจาก Sigmoid Layer เพื่อตั้ง Weight ให้อีกครั้งตามสมการที่ 2.5 และจะได้ออกมาเป็นค่า h(t)

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.5)$$

$$h_t = o_t * \tanh(C_t) \quad (2.6)$$



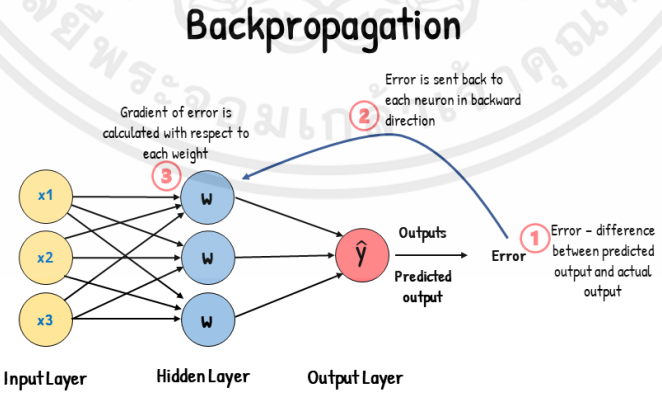
รูปที่ 2.11 องค์ประกอบของ Output gate layer

แหล่งที่มา : Sirawich Smitsomboon – Medium

ซึ่งมีการนำหลักการของโมเดลที่กล่าวมาใช้งานอย่างแพร่หลายในทางคอมพิวเตอร์วิทัศน์ การรู้จำเสียงพูด การประมวลผลภาษาธรรมชาติ การรู้จำเสียง และชีวสารสนเทศศาสตร์

2.3 โครงข่ายประสาทเทียมแบบแพร่ย้อนกลับ (Backpropagation)

โครงข่ายประสาทเทียมแบบแพร่ย้อนกลับเป็นวิธีการคำนวณ gradient ของ loss function ซึ่งสัมพันธ์กับ weights ของ network ด้วยการคำนวณจะเป็นการคำนวณไล่ไปทีละ layer ย้อนกลับมาจาก layer สุดท้ายเพื่อหลีกเลี่ยงการคำนวณที่ซ้ำซ้อนในเทอมกลางของ chain rule



รูปที่ 2.12 ตัวอย่าง Backpropagation

แหล่งที่มา : Gradient Descent vs. Backpropagation: What's the Difference? (analyticsvidhya.com)

2.3.1 การเรียนรู้ของโครงข่ายประสาทเทียมแบบแพร่ย้อนกลับ

การเรียนรู้ของโครงข่ายประสาทเทียมแบบแพร่ย้อนกลับ สามารถอธิบายได้ดังนี้

1. เตรียมข้อมูลสำหรับให้โมเดลเรียนรู้ จากนั้นกำหนดโหนดของ input layer กำหนดโหนดของ Hidden layer กำหนดโหนดของ Output layer กำหนดจำนวนชั้นของ Hidden layer กำหนดจำนวนรอบของการเรียนรู้ (Epoch) และค่าความผิดพลาดที่ยอมรับได้

2. กำหนดอัตราการเรียนรู้ (Learning Rate) ให้โมเดล

3. สุ่มค่าถ่วงน้ำหนักให้เส้นที่เชื่อมระหว่างโหนดในแต่ละชั้น ซึ่งค่านี้จะเป็นค่าจำนวนจริงที่มีค่าระหว่าง -1 ถึง 1 แต่สามารถอยู่ในช่วงอื่นๆได้ตามความเหมาะสมตามคุณสมบัติการออกแบบของโมเดล

4. นำเข้าข้อมูลสู่ input layer เพื่อให้โมเดลเรียนรู้

5. คำนวณค่า Output ที่ได้จากโมเดลและปรับค่าเวทของโมเดลจนกระทั่งได้ Output ถึงเป้าหมายที่กำหนดไว้ ซึ่งวิธีการคำนวณค่า Output มีดังนี้

การคำนวณไปข้างหน้า (Forward Pass) สมการที่ใช้ในการคำนวณหาผลรวมของ Input ที่เข้าไปยังโหนด j ใน Hidden layer ดังสมการ

$$x_j(p) = \sum_{i=1}^{n} x_i(p) w_{ij}(p) + b_j(p) w_i(p) \qquad (2.7)$$

เมื่อ p คือ จำนวนชุดข้อมูล

n คือ จำนวนโหนดทั้งหมดของ Input layer

$x_j(p)$ คือ ข้อมูลตัวอย่างที่เข้ามาที่ Input layer ตัวที่ i

$w_{ij}(p)$ คือ ค่าถ่วงน้ำหนักที่เชื่อมระหว่างข้อมูลตัวที่ i ไปยังโหนด j

$b_j(p)$ คือ ค่าไบอัสหรือค่าความเอนเอียง จะมีค่าเข้าใกล้ค่าเฉลี่ยจากการวัดหลายๆครั้งเมื่อเทียบกับค่าอ้างอิง

$w_i(p)$ คือ ค่าถ่วงน้ำหนักที่เชื่อมระหว่างไบอัสกับโหนด j

การคำนวณแบบแพร่ย้อนกลับ (Reverse Pass) จะทำการคำนวณหาค่าความคลาดเคลื่อนระหว่างผลลัพธ์ที่คำนวณได้จากโมเดลกับผลลัพธ์ที่ต้องการ แล้วจึงทำการส่งค่าความคลาดเคลื่อนดังกล่าวย้อนกลับมายังแต่ละโหนด โดยเริ่มตั้งแต่ Output layer แล้วส่งต่อไปยังชั้นต่างๆของโมเดล เมื่อทุกโหนดทราบถึงค่าความผิดพลาด จะนำค่าคลาดเคลื่อนดังกล่าวมาใช้ในการถ่วงน้ำหนัก โดยปรับค่าถ่วงน้ำหนักมากหรือน้อยขึ้นอยู่กับค่าความผิดพลาด

ค่าความคลาดเคลื่อนระหว่างผลลัพธ์ที่คำนวณได้กับผลลัพธ์ที่ต้องการ $(e_k(p))$ จากการเรียนรู้ของโมเดลที่ได้รับข้อมูลที่เตรียมไว้ เพื่อใช้ในการปรับค่าน้ำหนักใน Hidden layer สามารถคำนวณได้จาก

$$e_k(p) = y_{d,k}(p) - y_k(p) \qquad (2.8)$$

เมื่อ $y_{d,k}(p)$ คือ ค่าผลลัพธ์เป้าหมายที่ต้องการ

$y_k(p)$ คือ ค่าผลลัพธ์ที่คำนวณได้จากโมเดล

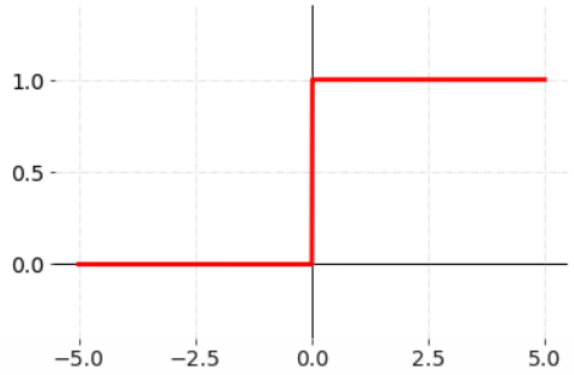
2.3.2 ฟังก์ชันกระตุ้น (Activation function)

ฟังก์ชันกระตุ้นหรือเรียกอีกชื่อว่า “ฟังก์ชันการส่งต่อ (Transfer function)” เป็นฟังก์ชันในการคำนวณเพื่อทำนายค่าของข้อมูลออก (Output) รูปแบบที่นิยมมากที่สุดและมีประโยชน์คือแบบที่ไม่เป็นฟังก์ชันสมการเส้นตรง (Non-linear function) เนื่องจากปัญหาในโลกความเป็นจริงมีลักษณะเป็นแบบสมการเส้นตรงน้อยมาก ฟังก์ชันกระตุ้นทำหน้าที่ในการตัดสินใจว่านิวรอนควรจะถูกกระตุ้นหรือไม่ โดยดูค่าผลรวมของข้อมูลเข้า (Input) และค่าน้ำหนัก (Weight) ฟังก์ชันกระตุ้นจะถูกนำไปใช้ทั้งโหนดซ่อน (Hidden node) และโหนดข้อมูลออก (Output node) ซึ่งทั้งสองโหนดอาจจะใช้ฟังก์ชันกระตุ้นที่เหมือนหรือต่างกันก็ได้ แต่ส่วนมากจะใช้ฟังก์ชันแบบไม่เป็นเชิงเส้น เนื่องจากในโหนดซ่อนจะมีการคำนวณแบบการรวมเชิงเส้น (Linear combination) ถ้าฟังก์ชันกระตุ้นของโหนดซ่อนจะมีการคำนวณแบบเชิงเส้นอีก จะเป็นการทำงานซ้ำซ้อนกับการคำนวณแบบการรวมเชิงเส้นในชั้นข้อมูลออก และจะทำให้ผลลัพธ์เทียบเท่ากับสมการถดถอยลอจิสติก

ฟังก์ชันกระตุ้นจะมีหลากหลายรูปแบบ ดังต่อไปนี้

1) ฟังก์ชันกระตุ้นค่าแบ่ง (Threshold Activation Function) ฟังก์ชันนี้เรียกอีกชื่อว่าฟังก์ชันไบนารี่สเตป (Binary step function) ซึ่งจะพิจารณาค่าข้อมูลเข้าว่ามากกว่าหรือน้อยกว่าค่าแบ่งที่กำหนดไว้ (threshold) หรือไม่เพื่อส่งค่าต่อไปยังชั้นถัดไป มีสมการดังต่อไปนี้

$$f(x) = \begin{cases} 0 & , x < 0 \\ 1 & , x > 0 \end{cases} \qquad (2.9)$$

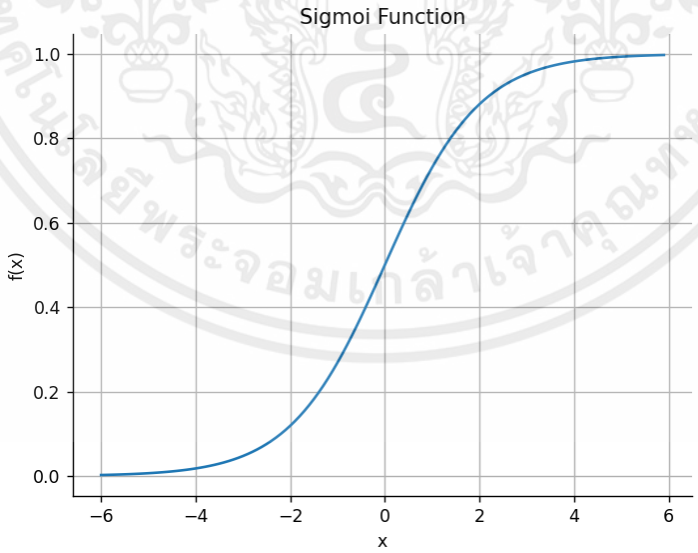


รูปที่ 2.13 กราฟแสดงฟังก์ชันไบนารี่สเตป

ที่มา: Getting to know Activation Functions in Neural Networks. | by Hasara Samson | Towards Data Science

2) ฟังก์ชั่นกระตุ้นซิกมอยด์ (Sigmoid Activation Function) เป็นฟังก์ชันทางคณิตศาสตร์ที่มีลักษณะเป็นตัวเอส “S-curve” หรือเรียกว่า “Sigmoid curve” จะมีค่าระหว่าง 0 และ 1 ฟังก์ชันนี้จะใช้เมื่อต้องการทำนายความน่าจะเป็น (Probability) ของข้อมูลออก มีสมการดังนี้

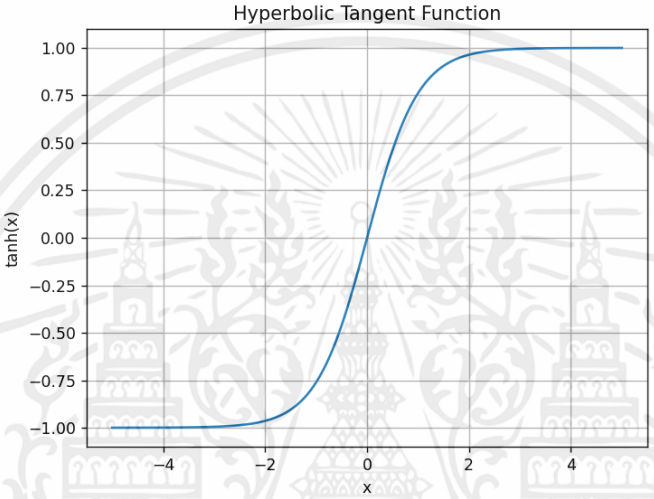
$$f(x) = \frac{e^x}{e^x+1} \qquad (2.9)$$



รูปที่ 2.14 กราฟแสดงฟังก์ชันซิกมอยด์

3) ฟังก์ชั่นไฮเปอร์โบลิคแทงก์เจนท์ (Hyperbolic Tangent Function: tanh) มีการทำงานคล้ายฟังก์ชันกระตุ้นซิกมอยด์แต่มีประสิทธิภาพดีกว่า จะมีค่าระหว่าง [-1, 1] ข้อดีของฟังก์ชันไฮเปอร์โบลิคแทงก์เจนท์ คือ สามารถแปลงค่าข้อมูลเข้าที่มีค่าเป็นลบมาก ๆ ให้เป็นข้อมูลออกที่ติดลบได้ และข้อมูลที่ค่าเป็นศูนย์จะถูกแปลงเป็นข้อมูลออกที่มีค่าใกล้ศูนย์ (near-zero output) มีสมการดังนี้

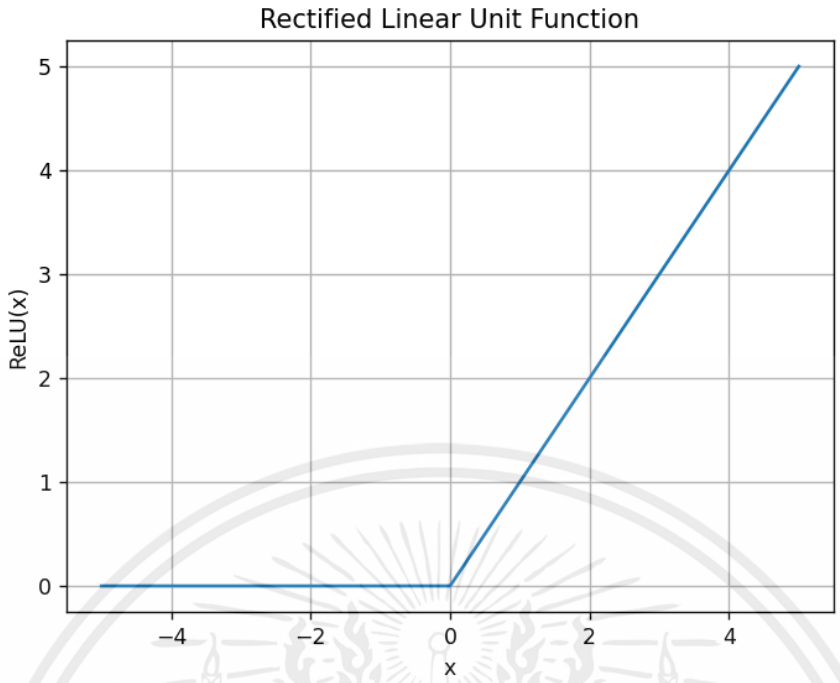
$$\tanh x = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (2.10)$$



รูปที่ 2.15 กราฟแสดงฟังก์ชันไฮเปอร์โบลิคแทงก์เจนท์

4) ฟังก์ชันเรคติไฟด์ลินเนียยูนิต (Rectified Linear Units, ReLu) เป็นฟังก์ชันที่นิยมใช้งานมากที่สุดในโครงข่ายประสาทเทียมแบบบิด (Convolutional Neural Networks: CNN) และโครงข่ายประสาทเทียมอัจฉริยะ (ANN) ฟังก์ชันนี้จะมีค่าอยู่ระหว่าง [, ∞) หมายถึงถ้าข้อมูลเข้ามีค่ามากกว่าศูนย์ข้อมูลออกจากเป็นค่าบวก และถ้าข้อมูลเข้ามีค่าศูนย์หรือติดลบ ข้อมูลออกจะมีค่าเป็นศูนย์ มีสมการดังนี้

$$ReLU(x) = \begin{cases} 0 & ,x < 0 \\ x & ,x > 0 \end{cases} \qquad (2.11)$$



รูปที่ 2.16 กราฟแสดงฟังก์ชันไฮเปอร์โบลิคแทงก์เจนท์

2.3.3 Optimizer

Optimizer คือ อัลกอลิทึมหรือวิธีการที่ถูกใช้เพื่อลดค่าของ Error function หรือ Loss function เป็นการเพิ่มประสิทธิภาพให้โมเดลมีความถูกต้องแม่นยำ Optimizer เป็นฟังก์ชันทางคณิตศาสตร์ที่ขึ้นอยู่กับพารามิเตอร์ในการเรียนรู้ของโมเดล เช่น Weights & Biases. Optimizer ช่วยในการปรับค่า weights และ learning rate ของโมเดลเพื่อลดค่า Loss

2.3.4 หลักการของค่าความผิดพลาด

ค่าความผิดพลาด หรือ Loss เป็นการวัดผลการทำงานของโมเดลในระดับย่อยๆ สำหรับข้อมูลแต่ละชิ้น เช่น ข้อมูลชิ้นที่ 1 ตอบผิดไปจากความเป็นจริง 10 แต้ม, ข้อมูลชิ้นที่ 2 ตอบผิดไป 2 แต้ม แต่ Accuracy เป็นการวัดผลการทำงานของโมเดลในภาพรวม เช่น โดยเฉลี่ยจากข้อมูลทั้งหมดแล้ว โมเดลนี้มีตอบถูกทั้งหมด 80%

โดยการนิยาม Loss นั้นอาจจะต่างกันไปสำหรับโมเดลที่ต่างกัน เช่น Cross-entropy loss สำหรับงาน Classification, MSE loss สำหรับงาน Regression, Wasserstein loss สำหรับ WGAN, Negative sampling loss สำหรับ word2vec เป็นต้น

2.3.5 Gradient Descent

Gradient Descent คือ Optimization algorithm ตัวหนึ่งสำหรับใช้ในการหาค่า weight ที่ทำให้โมเดลมีค่า Error หรือ Loss ต่ำที่สุดโดยมีสมการดังนี้

$$\theta_j = \theta_j - \propto \frac{\partial}{\partial \theta_j} J(\theta) \qquad (2.12)$$

โดย θ คือ Weight, $\propto$ คือ Learning rate และ $J(\theta)$ คือ Cost Function based on θ หรือ

- Learning rate คือ Hyperparameter ที่ควบคุมว่าจะเปลี่ยนแปลงค่า weight ของโมเดลใน 1 Step ของการเทรน ค่า Learning ส่งผลต่อการเทรนโมเดลโดยถ้าปรับมากเกินไปจะทำให้คำตอบของโมเดลไม่ลู่เข้าสู่คำตอบจริงหรือคำตอบที่ควรจะเป็น แต่ถ้าปรับน้อยเกินไปจะทำให้โมเดลของเราลู่เข้าสู่คำตอบช้าและใช้เวลาในการเทรนนานขึ้น อย่างไรก็ตามจำเป็นต้องมีการทดลองปรับค่า Learning Rate ว่าค่าไหนเหมาะสมกับโมเดล
- Cost Function หรือ Loss Function คือ ฟังก์ชันคณิตศาสตร์ที่คำนวณความผิดพลาด (Loss) ระหว่างคำตอบจริงกับการทำนายของโมเดล

New weight = weight – Learning rate * Gradient

Gradient Descent จะใช้สูตรในการคำนวณค่า weight ใหม่ที่ดีขึ้น โดยในรอบแรกของการคำนวณ Gradient Descent จะเริ่มจากการสุ่มเดาค่า weight ค่าแรกขึ้นมาก่อน และทำการวัด Performance ของโมเดลที่เกิดจากค่า weight ที่สุ่มขึ้นมา ด้วยค่า error จาก Cost Function และจะนำค่า error ที่ได้ มาช่วยในการคำนวณค่า weight ใหม่ที่ดีกว่าเดิม และนำค่า weight ปัจจุบันที่ได้มาทำแบบนี้วนซ้ำไปเรื่อย ๆ จนกว่าจะได้ค่า weight ที่ทำให้ค่า error ของ model ต่ำที่สุด ซึ่งการทำวนซ้ำของ Gradient Descent ในแต่ละครั้งก็คือการก้าวขยับเข้าไปหาจุดที่มีค่า error ต่ำที่สุดลงไปเรื่อย ๆ ดังนั้นระยะความยาวของการก้าวแต่ละครั้ง จึงส่งผลต่อความเร็วในการคำนวณ และถ้าก้าวยาวไปอาจจะทำให้ก้าวเลยจุดที่ดีที่สุดไปก็เป็นได้ ดังนั้นเราจึงต้องมีการกำหนดค่า Learning rate หรือ ระยะความยาวของการก้าวในแต่ละครั้งของ Gradient Descent ให้เหมาะสมด้วย Gradient Descent สามารถทำงานได้ทั้งในปัญหาประเภท regression และปัญหาประเภท classification โดยแค่ปรับเปลี่ยน Cost Function ตามประเภทของปัญหานั้น ๆ

สูตรของ Gradient ในปัญหาประเภท Regression จะมี Cost Function เป็น Mean Squared Error

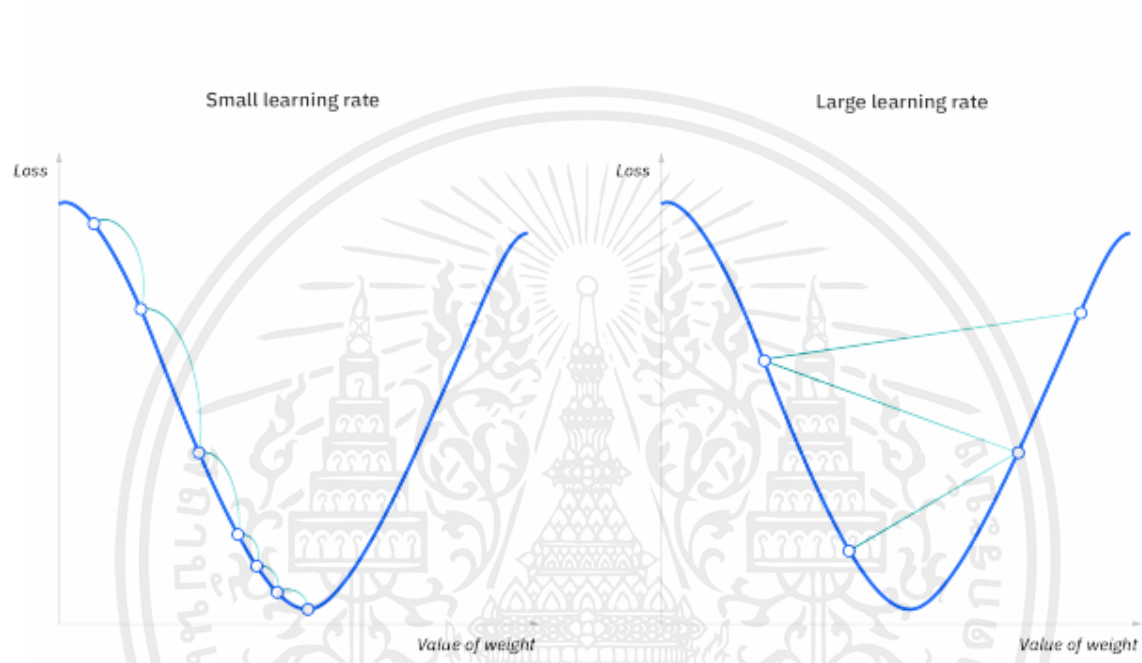
$$\text{Gradient} = \frac{\partial}{\partial \theta_j} \text{MSE}(\theta) \qquad (2.13)$$

โดย $\mathrm{MSE}(\theta)$ คือ Mean Squared Error

สูตรของ Gradient ในโจทย์ประเภท Classification จะมี Cost Function เป็น Cross-Entropy Loss

$$\text{Gradient} = \frac{\partial}{\partial \theta_j} \mathrm{L}(\theta) \qquad (2.14)$$

โดย $\mathrm{L}(\theta)$ คือ Cross-Entropy Loss



รูปที่ 2.17 อธิบาย Gradient Descent
แหล่งที่มา : What is Gradient Descent? | IBM

ประเภทของ Gradient descent

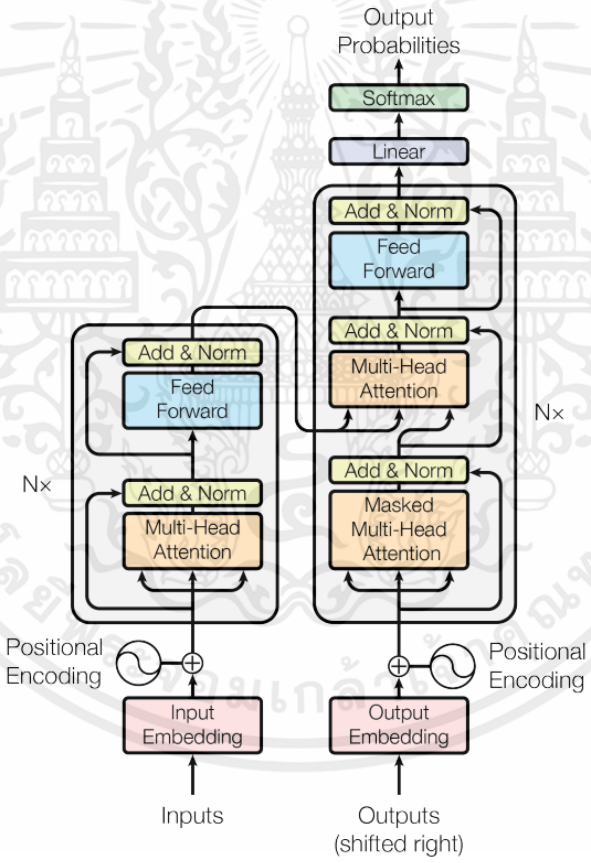
- Batch gradient descent คือ การนำค่าความผิดพลาด (Loss) ที่ได้จากรอบก่อนมาคำนวณหาผลลัพธ์ในรอบถัดไปโดยนำข้อทุกชุดมาคำนวณในแต่ละรอบ Batch gradient descent เป็นอัลกอริทึมที่มีประสิทธิในการลดค่า Loss แต่ใช้ระยะเวลาในการคำนวณมากเนื่องจากแต่ละรอบต้องคำนวณจากข้อมูลทุกชุด

- Stochastic gradient descent เป็นการคำนวณหาค่าความผิดพลาดเช่นเดียวกับ Batch gradient descent แต่ต่างกันตรงที่ Stochastic gradient descent จะสุ่มข้อมูลเพียงชุดเดียวมาคำนวณหาค่าความผิดพลาดในแต่ละรอบ ทำให้ใช้เวลาในการคำนวณน้อยกว่า Batch gradient descent แต่ค่า Loss ที่ได้จะลู่เข้าได้ไม่ดี ต้องใช้รอบในการคำนวณมากๆ

- Mini batch gradient descent เป็นการนำข้อดีของ Batch gradient descent และ Stochastic gradient descent มารวมกัน โดยแทนที่จะคำนวณข้อมูลทุกชุดแบบ Batch gradient descent หรือเพียงชุดเดียวแบบ Stochastic gradient descent. Mini batch Gradient Descent จะทำการคำนวณแต่ละรอบโดยใช้ข้อมูลจำนวนชุดตามที่เรากำหนด

2.4 Transformer

Transformer เป็นโมเดลที่ถูกออกแบบมาใช้สำหรับงานที่เป็น sequence to sequence สามารถแบ่งการทำงานออกได้เป็น 2 ส่วนคือ encoder และ decoder แต่ล่ะส่วนจะมี Multi-Head Attention เป็นหัวใจหลักในการทำงาน ตัวอย่างการใช้งาน Transformer เช่น การแปลภาษา จะรับ input ที่เป็น sequence และ Transformer จะสร้าง sequence ใหม่ขึ้นมาทางฝั่ง decoder



รูปที่ 2.18 โครงสร้างโมเดล Transformer

แหล่งที่มา : Attention is All you Need (neurips.cc)

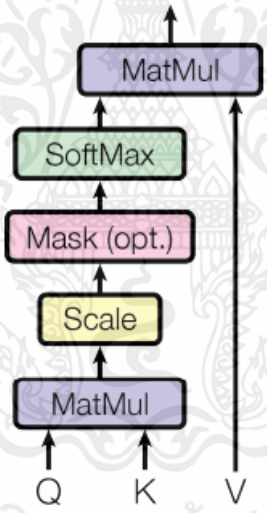
2.4.1 Self-Attention

Self-Attention คือ attention mechanisms ตัวหนึ่งที่ทำหน้าที่หาความสัมพันธ์ของ sequence ณ ตำแหน่งหนึ่งต่อ sequence ตำแหน่งอื่นเพื่อที่จะนำไปคำนวณเพื่อเปลี่ยนค่าของ sequence นั้นให้มีบริบทยิ่งขึ้นต่อไป

2.4.2 Scaled Dot-Product Attention

Scaled Dot-Product Attention เป็นกระบวนการทำ attention หนึ่งของ Self-Attention โดยมี input ประกอบด้วย queries (q) กับ keys (k) ในมิติ d_k และ values (v) ในมิติ d_v การทำงานเริ่มจากการคำนวณ dot products ระหว่าง q และทุกค่าของ k จากนั้นทำการ scale โดยหารด้วย $\sqrt{d_k}$ และใช้ฟังก์ชัน softmax เพื่อที่จะให้ได้ค่า weights ไปใช้กับ v ซึ่งเป็นไปดังสมการ

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \quad (2.14)$$



รูปที่ 2.19 ตัวอย่างโครงสร้าง Self-attention แบบ Scaled Dot-Product Attention
แหล่งที่มา : Attention is All you Need (neurips.cc)

2.4.3 Multi-Head Attention

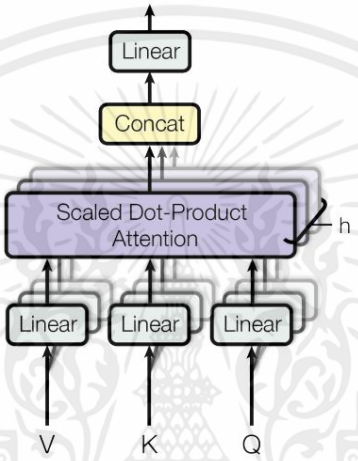
Multi-Head Attention เป็นโมดูลหนึ่งสำหรับ attention mechanisms ที่ทำงานโดยรัน attention mechanisms หลายตัวแบบขนาน outputs ของกระบวนการนี้จะถูกนำมาต่อกันแล้วทำการตัดสินใจให้เหลือแค่ชุดเดียว และ Multi-Head Attention เป็นโมดูลที่สำคัญในการทำงานตามโครงสร้างของ Transformer

$$\text{Multi Head}(Q, K, V) = [\text{head}_1, \ldots, \text{head}_h]W_0 \quad (2.15)$$

โดย $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

จากสมการข้างต้น W คือ weight ที่ได้มาจากการเรียนรู้ของ Multi-Head Attention

Multi-Head Attention โดยปกติแล้วจะถูกนำไปใช้กับงานประเภท sequence to sequence หรือ sequence to vector

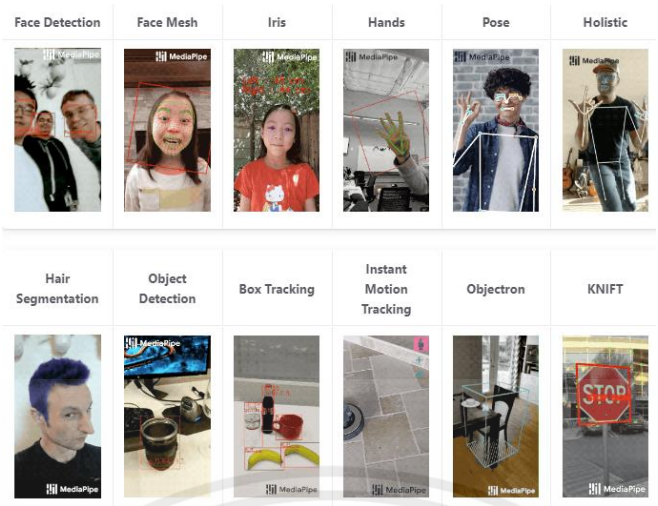


รูปที่ 2.20 โครงสร้างของ Multi-Head Attention ที่มี attention mechanisms เป็นแบบ Scaled Dot-Product Attention

แหล่งที่มา : Attention is All you Need (neurips.cc)

2.5 Media Pipe

Media Pipe เป็นแพลตฟอร์ม AI โดย Google แบบ Open source ที่สามารถใช้เป็น Pipeline ตรวจจับและรับรู้ใบหน้า มือ และท่าทางที่มีความซับซ้อน โดยใช้การเร่งความเร็วในการระบุและประมวลผล จึงออกมาเป็นโซลูชันที่แม่นยำและรวดเร็ว โซลูชันแบบครบวงจรใช้งานได้กับ Android, iOS, เดสก์ท็อป/คลาวด์, เว็บและ IoT



รูปที่ 2.21 ML solutions in MediaPipe

แหล่งที่มา : Home - mediapipe (google.github.io)

รูปที่ 2.22 ML Solutions และระบบที่รองรับ

แหล่งที่มา: Home – mediapipe (google.github.io)

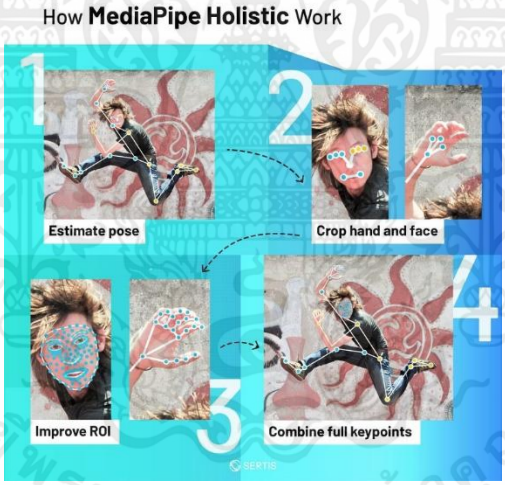
2.5.1 MediaPipe Holistic

MediaPipe Holistic คือ โซลูชั่นที่สามารถตรวจจับท่าทาง มือ และใบหน้าของมนุษย์ในเวลาเดียวกัน และเป็นโซลูชันแบบ end-to-end โซลูชันนี้จะใช้ Pipeline แบบใหม่ที่ประกอบด้วยการตรวจจับท่าทาง หน้า และมือที่ปรับแต่งให้ดีที่สุดเพื่อให้ทำงานได้เรียลไทม์ โดยใช้การโอนถ่ายหน่วยความจำระหว่าง Interference Backend ซึ่ง Pipeline จะรวมรูปแบบการปฏิบัติการและการประมวลผลที่แตกต่างกันตามการตรวจจับภาพแต่ละส่วนเข้าด้วยกัน และจะได้เป็นโซลูชันแบบครบวงจรที่ใช้งานได้แบบเรียลไทม์และสม่ำเสมอ

MediaPipe Holistic ใช้การทำงานแลกเปลี่ยนกันระหว่างการตรวจจับทั้งสามจุด โดยประสิทธิภาพของการทำงานจะขึ้นอยู่กับความรวดเร็วและคุณภาพของการแลกเปลี่ยนข้อมูล เมื่อรวมการตรวจจับทั้งสามเข้าด้วยกัน จะได้เป็นโซลูชันที่ทำงานร่วมกันเป็นหนึ่งเดียว โดยสามารถจับ Keypoints ของภาพเคลื่อนไหวได้ถึง 540+ จุด (ส่วนของท่าทาง 33 จุด มือข้างละ 21 จุด และส่วนใบหน้า 468 จุด)

1) การทำงานของ MediaPipe Holistic

MediaPipe Holistic ประมวลผลโดยการนำโมเดลของท่าทาง ใบหน้า และมือมารวมกัน ซึ่งทั้งสามส่วนได้รับการปรับคุณภาพให้เข้ากับโดเมนของตนเองที่สุด แต่เนื่องจากลักษณะการทำงานเฉพาะของสามส่วนที่ต่างกัน ทำให้ข้อมูลที่ใช้ได้ดีกับส่วนหนึ่งอาจไม่เข้ากับส่วนอื่น ยกตัวอย่างเช่น โมเดลการระบุท่าทาง อาจจะต้องการเฟรมวิดีโอที่มีความละเอียดที่ต่ำ แต่เมื่อต้องตัดส่วนของมือและหน้าจากภาพเพื่อส่งต่อไปยังโมเดลต่อไป ความละเอียดของภาพก็อาจจะต่ำเกินไปจนไม่สามารถประมวลผลได้แม่นยำ ด้วยเหตุนี้ MediaPipe Holistic จึงออกแบบมาในรูปแบบของ Pipeline ที่มีหลายขั้นตอน ซึ่งประมวลในแต่ละส่วนโดยใช้ความละเอียดภาพที่แตกต่างกัน



รูปที่ 2.23 การทำงานของ MediaPipe Holistic

แหล่งที่มา : MediaPipe Holistic อุปกรณ์ที่สามารถจับการเคลื่อนไหวของใบหน้า มือ และท่าทางได้ในเวลาเดียวกัน | by Sertis | Medium

อันดับแรก MediaPipe Holistic จะระบุท่าทางของมนุษย์โดยใช้โมเดลตรวจจับท่าทางและโมเดลระบุ Keypoint หลังจากนั้นจึงนำ Keypoint ที่ระบุได้มาแบ่งออกเป็น 3 จุดสนใจ (Region of Interest: ROI) ครอปตัดส่วนที่เป็นแขน 2 ข้าง และส่วนหน้า แล้วจึงใช้ส่วนที่ครอปออกมาแทนเพื่อเพิ่มความละเอียดของจุดนั้น จากนั้น Pipeline จะทำการครอปเฟรมที่มีความละเอียดสูงสุดของจุด ROI ทั้งสองจุด แล้วจึงใช้กับโมเดล

ที่ตรวจจับส่วนของใบหน้าและมือเพื่อระบุตำแหน่ง Keypoints ตามส่วนต่าง ๆ และขั้นตอนสุดท้ายจะเป็นการนำ Keypoint ที่ได้มารวมกับ Keypoint ของโมเดลท่าทางในตอนต้น รวมกันเป็น 540 keypoint+

การจะทำให้การระบุ ROI มีประสิทธิภาพขึ้นนั้น ต้องใช้ระบบตรวจจับแบบเดียวกับระบบที่ใช้ในอุปกรณ์ที่ตรวจจับใบหน้าหรือแขนเพียงอย่างเดียว ซึ่งจะใช้การอนุมานว่าวัตถุที่ตรวจจับไม่ได้มีการขยับมากนัก โดยจะใช้เฟรมก่อนหน้าเพื่อคาดการณ์การระบุตำแหน่งของวัตถุในเฟรมต่อไป อย่างไรก็ตาม ถ้าวัตถุขยับเร็วเกินไป ตัวติดตามตำแหน่งอาจผิดพลาด ซึ่งทำให้ตัวตรวจจับอาจต้องตรวจจับตำแหน่งในภาพใหม่อีกครั้ง

MediaPipe Holistic จะใช้การคาดการณ์ท่าทางในทุก ๆ เฟรมล่วงหน้าไว้เป็นเสมือนจุด ROI เสริมไว้ก่อนตั้งแต่แรกเพื่อลดระยะเวลาในการตอบสนองของ Pipeline เวลาที่พบการเคลื่อนไหวที่รวดเร็วเกินไป นอกจากนี้วิธีนี้ยังช่วยให้โมเดลสามารถรักษาความสอดคล้องกันได้ทั่วทั้งรูปร่างและป้องกันไม่ให้เกิดความสับสนระหว่างมือซ้ายและมือขวา หรือส่วนที่ต่างกันของร่างกายในแต่ละเฟรม

นอกจากนี้โดยปกติแล้วความละเอียดของเฟรมตรวจจับท่าทางนั้นจะต่ำเกินไปทำให้จุด ROI ของหน้าและมือนั้นมีความแม่นยำน้อยไป จนไม่สามารถให้แนวทางในการครอปตัดส่วนนั้นได้ ทำให้ต้องใช้โมเดลในการครอปตัดส่วนหน้าและมือที่มีขนาดเล็กแต่แม่นยำ เพื่อที่จะลดช่องว่างในเรื่องความแม่นยำระหว่างส่วนตัวและส่วนของมือกับใบหน้า โมเดลที่มีขนาดเล็กจะทำหน้าที่เป็นตัวแปลงพื้นที่ (Spatial Transformer) และยังใช้เวลาในการประมวลผลโมเดลน้อยลง 10 เปอร์เซ็นต์

2) ประสิทธิภาพการทำงาน

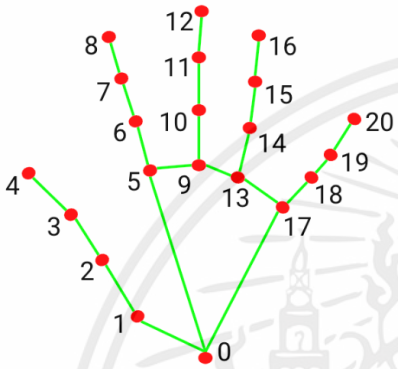
MediaPipe Holistic จำเป็นต้องใช้การทำงานร่วมกันระหว่าง 8 โมเดลต่อเฟรม แบ่งเป็น โมเดลตรวจจับท่าทาง 1 โมเดล โมเดลตรวจจับ Landmark ของท่าทาง 1 โมเดล โมเดล re-crop เพื่อครอปรูปใหม่ 3 โมเดล และโมเดลในการระบุ Keypoint สำหรับมือและใบหน้าอีก 3 โมเดล ซึ่งในระหว่างที่ Google พัฒนาโซลูชันนี้ พวกเขาได้ใช้ทั้ง Machine Learning และ Algorithm ในการคำนวณทั้งก่อนและหลังการประมวลผล ซึ่งโดยปกติแล้วการประมวลผลจะใช้เวลาค่อนข้างมากเนื่องจากความซับซ้อนของ Pipeline แต่ในกรณีของ MediaPipe Holistic พวกเขาได้ย้ายกระบวนการการคำนวณช่วงก่อนเริ่มดำเนินงานทั้งหมดไปไว้ที่ GPU ทำให้ Pipeline สามารถทำงานได้เร็วขึ้นโดยเฉลี่ย 1.5 เท่า แต่อาจจะแตกต่างกันไปบ้างในแต่ละอุปกรณ์ ดังนั้น MediaPipe Holistic จึงสามารถทำงานได้เกือบจะเรียลไทม์ แม้กระทั่งในอุปกรณ์ระดับกลางและในเบราว์เซอร์

คุณสมบัติของ Pipeline ที่ประกอบด้วยการทำงานหลายขั้นตอนนั้นช่วยเพิ่มประสิทธิภาพในการได้ใน 2 ส่วน หนึ่งคือเนื่องจากโมเดลส่วนมากเป็นโมเดลที่ทำงานแบบอิสระ จึงสามารถใช้โมเดลเวอร์ชั่นที่เล็กลงหรือใหญ่ขึ้นก็ได้ ขึ้นอยู่กับความแม่นยำและประสิทธิภาพที่ต้องการ หรือจะปิดโมเดลนั้นไปเลยก็ได้ และสองคือเมื่ออุปกรณ์สามารถตรวจจับท่าทางได้ ก็จะสามารถคาดเดาได้ว่ามือกับหน้าอยู่ในพื้นที่เฟรมที่เชื่อมต่อกันด้วยหรือไม่ ทำให้ Pipeline สามารถข้ามขั้นตอนการระบุส่วนเหล่านั้นไปได้

2.5.2 MediaPipe Hands

MediaPipe Hands เป็น solution การติดตามมือที่มีความเที่ยงตรงสูง ใช้ machine learning ในการสรุปจุดสังเกตุ 21 จุดในรูปแบบ 3 มิติของมือจากเฟรมเดียว

MediaPipe Hands ใช้ machine learning แบบทำงานร่วมกัน ได้แก่ palm detection model ที่เป็นโมเดลที่ทำงานในการตรวจจับฝ่ามือ และ hand landmark model ที่ทำงานบนภาพที่ถูกตัดจากขอบเขตของภาพที่ถูกตรวจพบโดย palm detection และส่งคืนค่าจุดสำคัญของมือแบบ 3 มิติที่มีความเที่ยงตรงสูง



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

รูปที่ 2.24 Position of hand landmarks

แหล่งที่มา : Hands - mediapipe (google.github.io)

2.5.2.1 Palm Detection Model

ขั้นแรก train palm detector แทน hand detector โดยการประมาณขอบเขตของวัตถุแข็งเช่นฝ่ามือหรือกำปั้น ซึ่งมีนัยสำคัญกว่าการไปตรวจจับทั้งมือที่มีส่วนประกอบของนิ้วมือเข้ามาด้วย เพิ่มเติมคือฝ่ามือนั้นเป็นวัตถุที่เล็กกว่า the non-maximum suppression algorithm จึงทำงานได้ดีแม้กระทั่งกรณีที่ทั้งสองมือบดบังกันเองอย่างเช่นการจับมือ ยิ่งไปกว่านั้นฝ่ามือสามารถออกแบบได้โดย square bounding boxes ละเว้นอัตราส่วนของภาพอื่นๆ จากเทคนิคที่กล่าวมาจะได้ความแม่นยำเฉลี่ย 95.7%

2.5.2.2 Hand Landmark Model

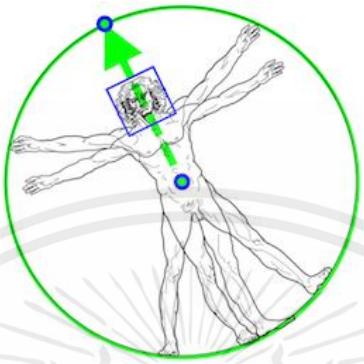
หลังจาก palm detection จากรูปทั้งหมดตามด้วย hand landmark model ดำเนินการตรวจจับตำแหน่งสำคัญของมือทั้ง 21 จุดแบบ 3 มิติ อย่างแม่นยำ

2.5.3 MediaPipe Pose

การประมาณค่าท่าทางของมนุษย์จากวิดีโอมีบทบาทสำคัญในการใช้งานต่างๆเช่นการหาปริมาณการออกกำลังกาย การรู้จำภาษามือ และการควบคุมด้วยท่าทางทั้งตัว

2.5.3.1 Person/pose Detection Model (BlazePose Detector)

คาดการณ์จุดสำคัญ 2 จุดเพื่อยืนยันจุดศูนย์กลางของร่างกายมนุษย์ การหมุน และขนาดในรูปแบบของวงกลม ทำการทำนายจุดศูนย์กลางที่ใช้คือจจุดกลางของสะโพกมนุษย์ รัศมีของวงกลมที่ล้อมรอบทั้งตัว และเส้นตรงมุมเอียงที่เชื่อมต่อระหว่างหัวไหล่กับจุดกลางของสะโพก

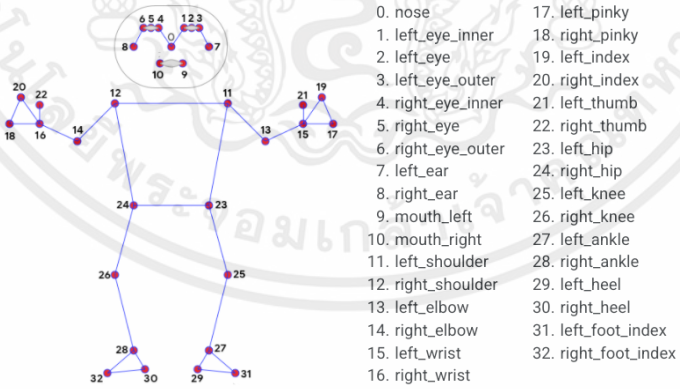


รูปที่ 2.25 Two virtual keypoints predicted by BlazePose detector in addition to the face bounding box.

แหล่งที่มา : Pose - mediapipe (google.github.io)

2.5.3.2 Pose Landmark Model (BlazePose GHUM 3D)

Landmark model ทำนายตำแหน่งของ pose landmarks ทั้ง 33 จุด



รูปที่ 2.26 Pose landmarks

แหล่งที่มา : Pose - mediapipe (google.github.io)

2.6 การวัดประสิทธิภาพในการจำแนกของโมเดล

Confusion Matrix เป็นเครื่องมือสำคัญในการประเมินผลลัพธ์ของการทำนายหรือ Prediction ของโมเดลที่ได้สร้างขึ้น โดยคำนวณจากการวัดว่าสิ่งที่โมเดลทำนาย (Prediction result) กับสิ่งที่เกิดขึ้นจริง (Actual result) มีสัดส่วนเป็นอย่างไร ซึ่งมีการกำหนดตัวแปรให้ค่าในตาราง confusion matrix ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 Confusion matrix

		Actual class: Positive	Actual class: Negative	
Predicted class	Positive	**TP**: True Positive	**FP:** False Positive (Type I Error)	**Precision:** $\frac{TP}{(TP + FP)}$
	Negative	**FN:** False Negative (Type II Error)	**TN:** True Negative	**Negative Predictive Value:** $\frac{TN}{(TN+FN)}$
		Recall or Sensitivity: $\frac{TP}{(TP + FN)}$	**Specificity:** $\frac{TN}{(TN + FP)}$	**Accuracy:** $\frac{TP + TN}{(TP + TN + FP + FN)}$

แหล่งที่มา : Performance Evaluation Measures of Classification model (analyticsvidhya.com)

True Positive (TP) คือค่าจากสิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้นจริง ในกรณีทำนายว่าจริงและสิ่งที่เกิดขึ้นก็คือ จริง

False Positive (FP) คือค่าจากสิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น ในกรณีทำนายว่าจริงแต่สิ่งที่เกิดขึ้นคือ ไม่จริง และถูกเรียกว่า Type I Error

True Negative (TN) คือค่าจากสิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น ในกรณีทำนายว่าไม่จริงและสิ่งที่เกิดขึ้นก็คือ ไม่จริง

False Negative (FN) คือค่าจากสิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง ในกรณีทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้นคือ จริง และถูกเรียกว่า Type II Error

หลังจากที่ได้อธิบายค่าในตาราง Confusion matrix ไปแล้วนั้น โมเดลที่มีประสิทธิภาพในการทำนายที่ดีควรที่จะมีค่า FN และ FP ที่น้อย แต่จะดีที่สุดหากค่า FN ในโมเดลนั้นน้อยที่สุดเนื่องจากค่า FN เป็นค่าที่โมเดลนั้นทำนายว่าจริง แต่ค่าที่ถูกต้องคือไม่จริง เราสามารถใช้ Confusion Matrix มาคำนวนการประเมินประสิทธิภาพของการทำนายด้วยโมเดลของเราในรูปแบบค่าต่างๆได้หลายค่า ได้แก่

2.6.1 Precision

เป็นการเปรียบเทียบการทำนายที่ถูกต้อง (Positive prediction) ว่า จริง และเกิดขึ้นจริง (TP) กับการทำนายว่า จริง แต่สิ่งที่เกิดขึ้น คือ ไม่จริง (FP) ซึ่งคำนวณได้จาก Precision = TP / (TP + FP)

2.6.2 Recall หรือ Sensitivity

ความถูกต้องของการทำนายว่าจะเป็น จริง เทียบกับจำนวนครั้งของเหตุการณ์ทั้งทำนาย และเกิดขึ้นว่า เป็นจริง คำนวณได้จาก Recall = TP/(TP+FN)

2.6.3 F1-Score

F1-Score เป็นค่าเฉลี่ยแบบ harmonic mean ระหว่าง precision และ recall จุดประสงค์ของการสร้าง F1 ขึ้นมา คือ เพื่อเป็น single metric ที่วัดความสามารถของโมเดล คำนวณได้จาก F1 = 2 x (Precision x Recall)/(Precision + Recall)

2.6.4 Accuracy

เป็นค่าความถูกต้องที่ทำนายได้ตรงกับสิ่งที่เกิดขึ้นจริงมากแค่ไหน หรือผลรวทของตัวเลขบนเส้นทแยงมุมในตาราง Confusion matrix โดย confusion matrix อาจเป็นตารางขนาด 3x3, 4x4, nxn ก็ได้ขึ้นกับจำนวน class ที่เราจะทำนาย ซึ่งคำนวณได้จาก Accuracy = (TP + TN) / (TP + FP +TN + FN)

2.7 Euclidean distance

Euclidean distance คือ ระยะทางปกติระหว่างจุดสองจุดในแนวเส้นตรง ซึ่งอาจสามารถวัดได้ด้วยไม้บรรทัด มีที่มาจากทฤษฎีบทพีทาโกรัส เหตุที่เรียกว่า แบบยุคลิด เนื่องจากเป็นการวัดระยะทางในปริภูมิแบบยุคลิด (หรือแม้แต่ปริภูมิผลคูณภายใน) คือไม่มีความโค้งและไม่สามารถทำให้โค้งงอ และการใช้สูตรนี้วัดระยะทางทำให้กลายเป็นปริภูมิอิงระยะทาง ค่าประจำ (norm) ที่เกี่ยวข้องก็จะเรียกว่าเป็น ค่าประจำแบบยุคลิด (Euclidean norm) เช่นกัน (งานเขียนสมัยก่อนเรียกการวัดอย่างนี้ว่า ระยะทางแบบพีทาโกรัส)

2.7.1 นิยามของ Euclidean distance

ระยะทางแบบยุคลิดระหว่างจุดสองจุด p และ q คือความยาวของส่วนของเส้นตรง pq ถ้า p = (p1, p2, ..., pn) และ q = (q1, q2, ..., qn) ในระบบพิกัดคาร์ทีเซียน เป็นจุดสองจุดบนปริภูมิยุคลิด n มิติ ระยะทางระหว่างจุด p กับ q คำนวณได้จาก

$$d(p,q) = \sqrt{(p_1 - q_1)^2+(p_2 - q_2)^2+...+ (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \qquad (2.16)$$

ค่าประจำแบบยุคลิด คือระยะทางจากจุดหนึ่งจุด p ไปยังจุดกำเนิด (0, 0, ..., 0) บนปริภูมิยุคลิด

$$\|p\| = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} \qquad (2.17)$$

ซึ่งสมการตัวหลังเกี่ยวข้องกับผลคูณจุด เป็นขนาดของเวกเตอร์ p จากจุดกำเนิด ระยะทางแบบยุคลิดจึงอาจนิยามได้อีกแบบหนึ่งดังนี้

$$\|p - q\| = \sqrt{(p - q) \cdot (p - q)} = \sqrt{\|p\|^2 + \|q\|^2 - 2p \cdot q} \qquad (2.18)$$

ในหนึ่งมิติ ระยะทางระหว่างจุดสองจุดบนเส้นจำนวนจริงคือค่าสัมบูรณ์ของผลต่างของสองค่านั้น ดังนั้นถ้าให้ p และ q เป็นจุดสองจุด (หรือจำนวนสองจำนวน) บนเส้นจำนวนจริงแล้ว ระยะทางระหว่าง p และ q จึงคำนวณได้จาก

$$d(p,q) = \sqrt{(p_1 - q_1)^2} = |p - q| \qquad (2.19)$$

ในสองมิติแบบยุคลิด ถ้า p = (p1, p2) และ q = (q1, q2) แล้ว ระยะทางระหว่าง p และ q สามารถคำนวณได้ดังนี้ ซึ่งมีสูตรเหมือนกับทฤษฎีบทพีทาโกรัส

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \qquad (2.20)$$

จากนิยามแบบที่สองของระยะทางแบบยุคลิด ถ้าหาก p = (r1, θ1) และ q = (r2, θ2) ในระบบพิกัดเชิงขั้ว จะสามารถคำนวณระยะทางได้จากสูตรนี้

$$\|p - q\| = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta_1 - \theta_2)} \qquad (2.21)$$

ในสามมิติแบบยุคลิด ระยะทางระหว่าง p และ q ก็คือ

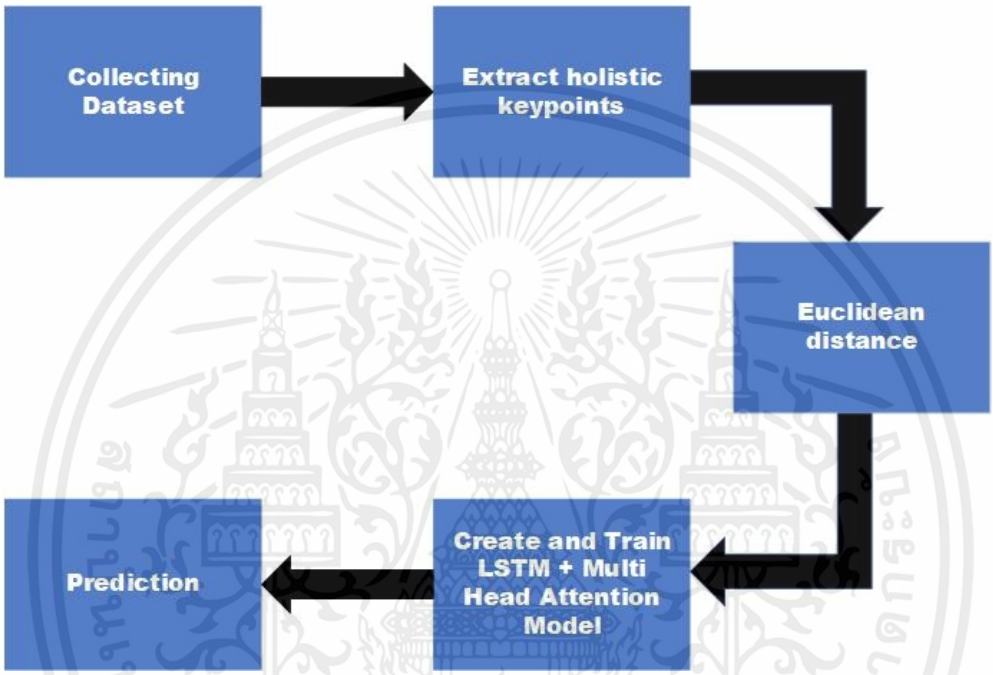
$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2} \qquad (2.22)$$

เมื่อมิติเพิ่มขึ้น พจน์ภายในก็เพิ่มขึ้นตามจำนวนมิติ เช่นนี้เรื่อยไป

บทที่ 3

หลักการทำงานและการออกแบบ

3.1 System block diagram



รูปที่ 3.1 System block diagram

3.1.1 Collecting Data

ในขั้นตอนแรกทำการเก็บข้อมูลท่าทางภาษามือ ในที่นี้เก็บทั้งหมด 51 ท่า ท่าละ 120 วิดีโอ วิดีโอ 60 เฟรม รวมทั้งหมด 6,120 วิดีโอ แบ่งเป็นข้อมูลสำหรับให้โมเดลเรียนรู้ 4,896 วิดีโอ สำหรับทดสอบ 1,224 วิดีโอ

3.1.2 Extract holistic keypoints

จาก MediaPipe Holistic สามารถดึง keypoints ของมือ และท่าทางได้ดังนั้นจึงใช้ทำการดึง landmarks ของมือทั้งสองข้างและ landmarks ของท่าทางทางบนร่างกายออกมาเพื่อนำไปใช้คำนวณ Euclidean distance

3.1.3 Euclidean distance

3.1.3.1 Hand landmarks

ดึง landmarks ของมือออกมาทั้งหมดมือล่ะ 11 จุดได้แก่ ปลายนิ้วของนิ้วทั้ง 5 ข้อต่อตรงกลางของนิ้วทั้ง 5 และฝ่ามือ 1 จุดซึ่งเป็นจุดอ้างอิงเนื่องจากมีความเปลี่ยนแปลงน้อยที่สุดเมื่อเทียบกับ landmarks จุดอื่น จากนั้นนำแต่ล่ะจุดจากทั้ง 10 จุดมาหา Euclidean distance กับฝ่ามือที่เป็นจุดอ้างอิง

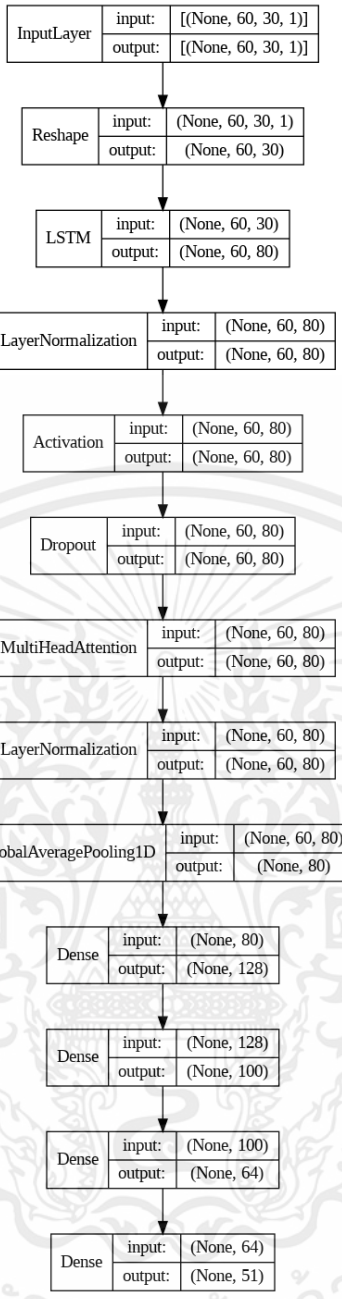
3.1.3.2 Pose landmarks

ดึง landmarks ของท่าทางในส่วนของแขนออกมาทั้งหมดข้างล่ะ 6 จุดได้แก่ นิ้วโป้ง นิ้วชี้ นิ้วก้อย ฝ่ามือ ข้อศอก และหัวไหล่ โดยให้หัวไหล่เป็นจุดอ้างอิง จากนั้นนำแต่ล่ะจุดจากทั้ง 2 จุดมาหา Euclidean distance กับหัวไหล่ที่เป็นจุดอ้างอิง

3.1.4 Create and train LSTM + Multi Head model

3.1.4.1 Deep Learning

สร้าง Deep learning model จากโมดูล LSTM ตามด้วยโมดูล Multi-Head Attention โดยนำ features ที่เป็น sequence มาเข้า input layer จากนั้นส่งต่อไปที่โมดูล LSTM ที่มี 80 units เพื่อให้โมเดลเรียนรู้ข้อมูลที่เป็น sequence ต่อไปทำการ normalize layer ด้วยโมดูล LayerNormalization เพื่อเพิ่มประสิทธิภาพในการลู่เข้าของ gradient จาก Relu activation function ให้ดียิ่งขึ้น จากนั้นส่งต่อข้อมูลไปยังโมดูล Dropout เพื่อทำการลบข้อมูลบางส่วนตามอัตราส่วนที่กำหนดในขณะที่โมเดลกำลังเรียนรู้ ซึ่งการทำแบบนี้จะช่วยลดโอกาสการ overfit ของโมเดลได้ จากนั้นส่งต่อข้อมูลไปยังโมดูล Multi-Head Attention โมดูลนี้จะทำการหาความสัมพันธ์กันระหว่างทุก sequence จาก sequence ทั้งหมด และเพิ่มประสิทธิภาพการลู่เข้าของ gradient ด้วยโมดูล LayerNormalization จากนั้นจัดการข้อมูลก่อนนำไปสู่การตัดสินใจของโมเดลโดยการนำ sequence ทั้งหมดมารวมค่าเฉลี่ยด้วยโมดูล GlobalAveragePooling1D และสุดท้ายโมเดลตัดสินใจ classify ข้อมูลด้วย Dense layer



รูปที่ 3.2 โครงสร้าง Deep learning model

3.1.5 Prediction

ทำการทำนายและหาประสิทธิภาพในการจำแนกคำแปลของโมเดลว่ามีความแม่นยำมากแค่ไหน ซึ่งในโครงงานนี้เราจะใช้เครื่องมือดังนี้

3.1.5.1 Confusion Matrix

ใช้เครื่องมือ confusion matrix จาก library ของ sklearn แล้วทำการกำหนดพารามิเตอร์ระหว่างค่าจริงกับค่าที่ทำนายได้จากโมเดล confusion_matrix(actual, predicted) จะได้เป็นตาราง confusion matrix ออกมาขนาดเท่ากับจำนวน class คูณจำนวน class ที่เรากำหนด และจะใช้เครื่องมือนี้กับโมเดล deep learning ที่ได้สร้างขึ้น คือ LSTM และ Multihead attention

3.1.5.2 Accuracy

ใช้เครื่องมือ accuracy score จาก library ของ sklearn แล้วทำการกำหนดพารามิเตอร์ระหว่างค่าจริงกับค่าที่ทำนายได้จากโมเดล accuracy_score(actual,predicted) จะได้ค่าออกมาเป็นค่าร้อยละของความแม่นยำของโมเดล deep learning ที่สร้างขึ้นมา

3.1.5.3 Precision, Recall หรือ Sensitivity, F1-Score

ใช้เครื่องมือ classification report จาก library ของ sklearn แล้วกำหนดพารามิเตอร์ระหว่างค่าจริงกับค่าที่ทำนายได้จากโมเดล และกำหนด output_dict=True เพื่อจัดข้อมูลให้อยู่ในรูปแบบ dictionary จะได้ค่า Precision, Recall หรือ Sensitivity, F1-Score ของแต่ละคำออกมาเพื่อมาดูประสิทธิภาพ

บทที่ 4
การทดลองและผลการทดลอง

4.1 คุณสมบัติของโปรแกรมแปลภาษามือ

1). สามารถแสดงความหมายของภาษามือได้ 51 ท่าดังนี้ สบายดี, เด็ก, สวัสดี, ขอโทษ, ขยั่น, เมื่อวาน , เข้าใจ, พวกเรา, เศร้า, ขอบคุณ, ง่าย, พวกเขา, ชื่อ, ที่ไหน, รัก, ต้องการ, ทำ, ผู้ชาย, อิ่ม, หัวเราะ, กิน, ชอบ, มา, ไม่เข้าใจ, ร้องไห้, งาน, ทำไม, คิดถึง, แม่, ยิ้ม, พรุ่งนี้, ฉัน, อะไร, หิว, สงสัย, โกรธ, กลัว, คุณ, ขอบคุณมาก, พ่อ, ขี้เกียจ, ไป, เมื่อไร, ผู้หญิง, ไม่ชอบ, ลืม, มีความสุข, วันนี้, ยาก, อย่างไรและจำได้

2). สามารถแสดงความหมายของภาษามือออกมาเป็นประโยคได้ตามที่ต้องการ

3). รองรับการใช้งานบนระบบปฏิบัติการ Windows

4.2 การทดลอง

1). Train โมเดล LSTM แล้วนำไปเข้ากับโมเดล Multi-head attention

2). Predict test dataset ด้วยโมเดล LSTM + Multi-head attention

3). ประเมิณผลโมเดล

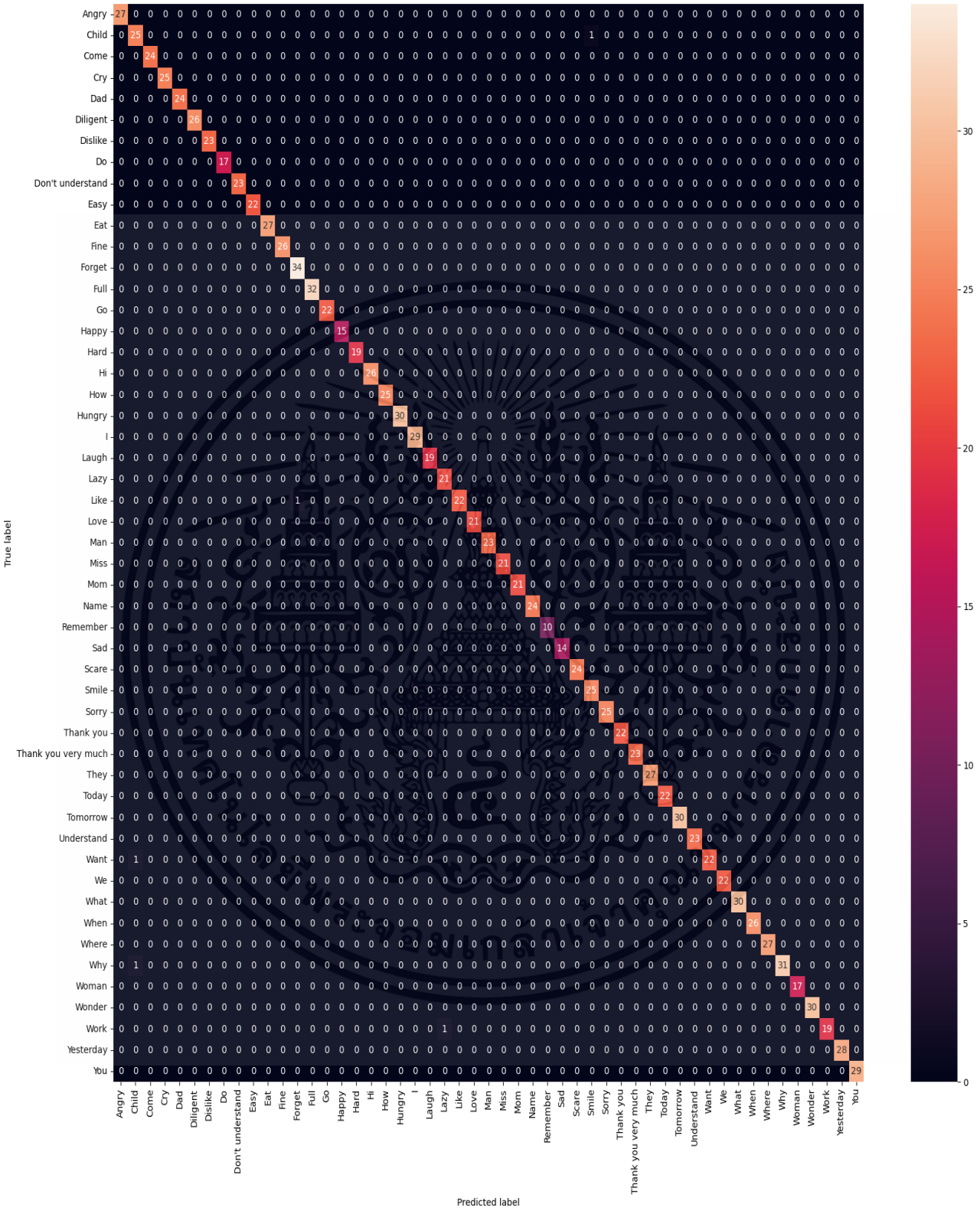
4.3 ผลการทดลอง

4.3.1 Accuracy score

ตารางที่ 4.1 แสดงค่า Accuracy score ของ model

	Accuracy score
LSTM + Multi-head attention	0.9828

4.3.2 Confusion matrix



รูปที่ 4.1 Confusion matrix ของโมเดล LSTM + Multi-head attention

4.3.3 Classification report

ตารางที่ 4.2 LSTM + Multi-head attention

	สบายดี	เด็ก	สวัสดี	ขอโทษ	ขยัน	เมื่อวาน	เข้าใจ	พวกเรา
Precision	1	0.862069	1	0.961538	1	1	0.956522	1
Recall	1	0.961538	1	1	1	1	0.956522	0.954545
f1-score	1	0.909091	1	0.980392	1	1	0.956522	0.976744
Support	26	26	26	25	26	28	23	22

	เศร้า	ขอบคุณ	ง่าย	พวกเขา	ชื่อ	ที่ไหน	รัก	ต้องการ
Precision	0.933333	1	1	0.931034	1	1	0.913043	1
Recall	1	0.954545	1	1	1	1	1	0.913043
f1-score	0.965517	0.976744	1	0.964286	1	1	0.954545	0.954545
Support	14	22	22	27	24	27	21	23

	ทำ	ผู้ชาย	อิ่ม	หัวเราะ	กิน	ชอบ	มา	ไม่เข้าใจ
Precision	1	1	1	1	0.964286	0.954545	1	1
Recall	1	1	0.96875	0.894737	1	0.913043	1	0.956522
f1-score	1	1	0.984127	0.944444	0.981818	0.933333	1	0.977778
Support	17	23	32	19	27	23	24	23

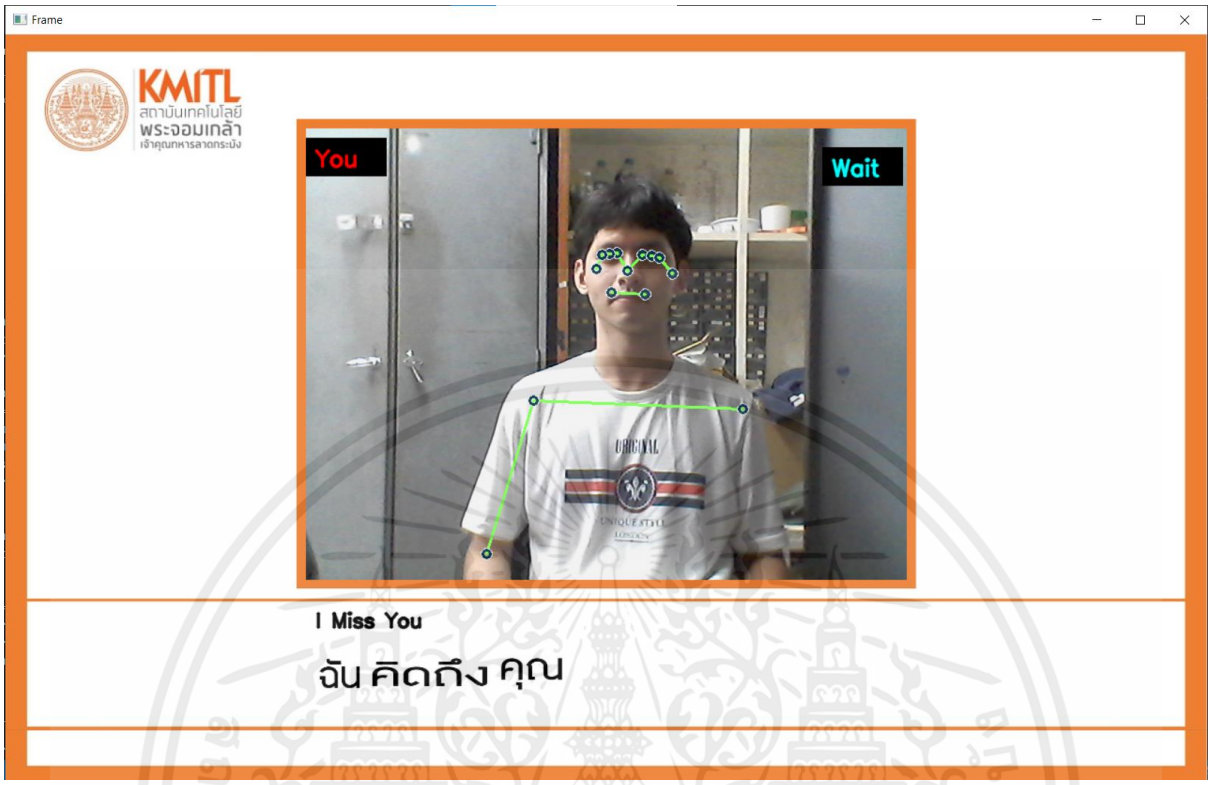
	ร้องไห้	ทำงาน	ทำไม	คิดถึง	แม่	ยิ้ม	พรุ่งนี้	ฉัน
Precision	1	1	1	1	1	0.961538	1	1
Recall	1	0.95	0.90625	1	1	1	1	0.896552
f1-score	1	0.974359	0.95082	1	1	0.980392	1	0.945455
Support	25	20	32	21	21	25	30	29

	อะไร	หิว	สงสัย	โกรธ	กลัว	คุณ	ชอบคุณมาก	พ่อ
Precision	1	0.966667	1	0.931034	1	1	1	0.96
Recall	0.966667	0.966667	1	1	1	1	1	1
f1-score	0.983051	0.966667	1	0.964286	1	1	1	0.979592
Support	30	30	30	27	24	29	23	24

	ขี้เกียจ	ไป	เมื่อไร	ผู้หญิง	ไม่ชอบ	ลืม	มีความสุข	วันนี้
Precision	0.954545	1	1	1	1	0.971429	1	0.956522
Recall	1	1	1	1	1	1	1	1
f1-score	0.976744	1	1	1	1	0.985507	1	0.977778
Support	21	22	26	17	23	34	15	22

	ยาก	อย่างไร	จำได้
Precision	1	1	1
Recall	1	1	1
f1-score	1	1	1
Support	19	25	10

4.3.4 Real time prediction



รูปที่ 4.2 Real time prediction and display in sentence

บทที่ 5

สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดสอบโมเดล LSTM + Multi-head attention พบว่ามีค่า accuracy score อยู่ที่ประมาณ 0.9828 หรือ 98.28 % จาก confusion matrix พบว่าโมเดลมีการทำนายผิดพลาดน้อยมาก จาก classification report จะพบว่าสำหรับ precision ค่าที่ต่ำที่สุดมีค่าประมาณ 0.862 ที่คำว่า child สำหรับ recall ค่าที่ต่ำที่สุดมีค่าประมาณ 0.895 ที่คำว่า laugh สำหรับ f1-score มีค่าต่ำที่สุดประมาณ 0.909 ที่คำว่า child

5.2 วิจารณ์ผลการทดลอง

5.2.1 จากการทดสอบแบบ real time prediction พบว่าโมเดลนี้มีการทำนายได้แม่นยำมากแต่ขาดความยืดหยุ่นในทางปฏิบัติ เนื่องจากการมีหลายคลาสซึ่งบางคลาสมีลักษณะการทำท่าทางที่คล้ายคลึงกันมาก จึงจำเป็นที่จะต้องทำท่าให้ใกล้เคียงกับข้อมูลที่ทำการเทรนให้โมเดลมากที่สุดเพื่อที่จะให้โมเดลทำนายคลาสออกมาได้อย่างถูกต้อง

5.2.2 ในการทำนายแต่ละคลาสจะต้องใช้เวลาในการทำท่าท่าละ 60 เฟรม ทำให้การนำคำที่ทำนายเสร็จแล้วมาเรียงประกอบกันให้เป็นประโยคเป็นไปโดยไม่ต่อเนื่องลื่นไหลเท่าที่ควร

5.2.3 การทำท่าควรที่จะให้โปรแกรมแท็กจุดบนมือและส่วนอื่นๆที่จะนำไปใช้ในการทำนายให้ชัดเจนเพื่อความแม่นยำของการทำนาย

5.2.4 สำหรับโมเดล LSTM + Multi-head attention นี้เป็นโมเดลที่ใช้สำหรับงานที่รับข้อมูล input เป็น sequence เข้าไป จากนั้นจะได้ output ที่เป็น vector ออกมา แล้วนำไป classify ออกมาเป็นคำสุดท้ายนำคำแต่ละคำมาต่อเรียงกันให้เป็นประโยค แต่มีความเป็นไปได้ที่จะพัฒนาโมเดลต่อไปเป็น Transformer โดยการสร้างส่วนที่เป็น decoder เพิ่มจากโมเดลเดิมที่เป็น encoder อยู่ก่อนแล้ว ซึ่งโมเดล Transformer นี้จะสามารถรับข้อมูล input ที่เป็น sequence และสร้าง output เป็น sequence ได้ ซึ่งจะเพิ่มความแม่นยำของการสร้างประโยคได้จากการทำงานเต็มประสิทธิภาพของ attention mechanism

เอกสารอ้างอิง

[1] Vithan Minaphinant. (2018). Machine Learning คืออะไร?. Available at: Machine Learning คืออะไร?. เคยสงสัยกันบ้างรึเปล่า ว่า | by Vithan Minaphinant | investic | Medium

[2] IT Arena. (2015). Introduction to Deep Learning (Dmytro Fishman Technology Stream). Available at: Introduction to Deep Learning (Dmytro Fishman Technology Stream) (slideshare.net)

[3] Wikipedia. (2022). Cluster analysis. Available at: Cluster analysis - Wikipedia

[4] Wikipedia. (2022). Reinforcement learning. Available at: Reinforcement learning - Wikipedia

[5] Wikipedia. (2021). โครงข่ายประสาทเทียม. Available at: โครงข่ายประสาทเทียม - วิกิพีเดีย (wikipedia.org)

[6] Sirinart Tangruamsub. (2017). Long Short-Term Memory (LSTM). Available at: Long Short-Term Memory (LSTM). คิดว่าหลายๆ คนที่เคยทำ machine learning... | by Sirinart Tangruamsub | Medium

[7] Sirawich Smitsomboon. (2020). สรุปความเข้าใจ RNN, LSTM, GRU (24/10/2020). Available at: สรุปความเข้าใจ RNN, LSTM, GRU (24/10/2020) | by Sirawich Smitsomboon | Medium

[8] Wikipedia. (2023). Backpropagation. Available at: Backpropagation - Wikipedia

[9] Mbali Kalirane. (2023). Gradient Descent vs. Backpropagation: What's the Difference?. Available at: Gradient Descent vs. Backpropagation: What's the Difference? (analyticsvidhya.com)

[10] Hasara Samson. (2020). Getting to know Activation Functions in Neural Networks. Available at: Getting to know Activation Functions in Neural Networks. | by Hasara Samson | Towards Data Science

[11] IBM. What is gradient descent?. Available at: What is Gradient Descent? | IBM

[12] Wikipedia. (2021). Self-attention. Available at: Self-attention - Wikipedia

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser and Illia Polosukhin. (2017). Attention Is All You Need. Available at: Attention is All you Need (neurips.cc)

[14] Vaswani et al. Multi-Head Attention. Available at: Multi-Head Attention Explained | Papers With Code

[15] Sertis. (2021). MediaPipe Holistic อุปกรณ์ที่สามารถจับการเคลื่อนไหวของใบหน้า มือ และท่าทางได้ในเวลาเดียวกัน. Available at: MediaPipe Holistic อุปกรณ์ที่สามารถจับการเคลื่อนไหวของใบหน้า มือ และท่าทางได้ในเวลาเดียวกัน | by Sertis | Medium

[16] Anjana Ambika. (2020). Decluttering the performance measures of classification models. Available at: Performance Evaluation Measures of Classification model (analyticsvidhya.com)

[17] NMP Channel. (2019). ภาษามือไทยขั้นพื้นฐาน (THsl Basic01). Available at: https://www.youtube.com/watch?v=ATcM_kNgbcM

[18] NMP Channel. (2019). ภาษามือไทยขั้นพื้นฐาน (THsl Basic#04). Available at: https://www.youtube.com/watch?v=PdRvh1-Km5c

[19] นกเอี้ยง เสียงทอง. (2020). เฮฮาภาษามือ EP8 ตอน(คำศัพท์ที่ใช้สื่อสารในชีวิตประจำวัน). Available at: https://www.youtube.com/watch?v=uBAxJCTL6L0

[20] ZEN Group People. (2020). ภาษามือในที่ทำงาน By ZEN Group : EP1. Available at: https://www.youtube.com/watch?v=JAAj3SP9nbw

ภาคผนวก

Source code (Python)

```
# Install and Import dependencies tool
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import mediapipe as mp
import seaborn as sns
import pandas as pd
from time import sleep
import mediapipe as mp
```

```
# Create a function to draw and extract keypoints
mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities

def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2
RGB
    image.flags.writeable = False                  # Image is no longer
writeable
    results = model.process(image)                 # Make prediction
    image.flags.writeable = True                   # Image is now writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # COLOR COVERSION RGB 2 BGR
    return image, results

def draw_styled_landmarks(image, results):

    # Draw pose connections
    mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(80,55,10),
thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(80,255,121),
thickness=2, circle_radius=2)
                              )
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76),
thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(17,17,250),
thickness=2, circle_radius=2)
                              )
    # Draw right hand connections
```

```
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,17,17),
thickness=2, circle_radius=2)
                              )
```

```
def lh_euclid(ref_0, thumb_3, thumb_4, index_6, index_8, mid_10, mid_12,
ring_14, ring_16, pinky_18, pinky_20):
    thumb_3_list = []
    thumb_4_list = []
    index_6_list = []
    index_8_list = []
    mid_10_list = []
    mid_12_list = []
    ring_14_list = []
    ring_16_list = []
    pinky_18_list = []
    pinky_20_list = []
    # euclidean thumb fing
    euclid_thumb_3 = np.sqrt((thumb_3.x - ref_0.x)**2 + (thumb_3.y -
ref_0.y)**2 + (thumb_3.z - ref_0.z)**2)
    thumb_3_list.append(euclid_thumb_3)
    euclid_thumb_4 = np.sqrt((thumb_4.x - ref_0.x)**2 + (thumb_4.y -
ref_0.y)**2 + (thumb_4.z - ref_0.z)**2)
    thumb_4_list.append(euclid_thumb_4)
    # euclidean index fing
    euclid_index_6 = np.sqrt((index_6.x - ref_0.x)**2 + (index_6.y -
ref_0.y)**2 + (index_6.z - ref_0.z)**2)
    index_6_list.append(euclid_index_6)
    euclid_index_8 = np.sqrt((index_8.x - ref_0.x)**2 + (index_8.y -
ref_0.y)**2 + (index_8.z - ref_0.z)**2)
    index_8_list.append(euclid_index_8)
    # euclidean mid fing
    euclid_mid_10 = np.sqrt((mid_10.x - ref_0.x)**2 + (mid_10.y - ref_0.y)**2
+ (mid_10.z - ref_0.z)**2)
    mid_10_list.append(euclid_mid_10)
    euclid_mid_12 = np.sqrt((mid_12.x - ref_0.x)**2 + (mid_12.y - ref_0.y)**2
+ (mid_12.z - ref_0.z)**2)
    mid_12_list.append(euclid_mid_12)
    # euclidean ring fing
    euclid_ring_14 = np.sqrt((ring_14.x - ref_0.x)**2 + (ring_14.y -
ref_0.y)**2 + (ring_14.z - ref_0.z)**2)
    ring_14_list.append(euclid_ring_14)
    euclid_ring_16 = np.sqrt((ring_16.x - ref_0.x)**2 + (ring_16.y -
ref_0.y)**2 + (ring_16.z - ref_0.z)**2)
    ring_16_list.append(euclid_ring_16)
```

```
    # euclidean pinky fing
    euclid_pinky_18 = np.sqrt((pinky_18.x - ref_0.x)**2 + (pinky_18.y -
ref_0.y)**2 + (pinky_18.z - ref_0.z)**2)
    pinky_18_list.append(euclid_pinky_18)
    euclid_pinky_20 = np.sqrt((pinky_20.x - ref_0.x)**2 + (pinky_20.y -
ref_0.y)**2 + (pinky_20.z - ref_0.z)**2)
    pinky_20_list.append(euclid_pinky_20)
    # concatenate
    lh_euclid = np.concatenate([thumb_3_list, thumb_4_list, index_6_list,
index_8_list, mid_10_list, mid_12_list,
                                ring_14_list, ring_16_list,
pinky_18_list,pinky_20_list])

    return lh_euclid
def rh_euclid(ref_0, thumb_3, thumb_4, index_6, index_8, mid_10, mid_12,
ring_14, ring_16, pinky_18, pinky_20):
    thumb_3_list = []
    thumb_4_list = []
    index_6_list = []
    index_8_list = []
    mid_10_list = []
    mid_12_list = []
    ring_14_list = []
    ring_16_list = []
    pinky_18_list = []
    pinky_20_list = []
    # euclidean thumb fing
    euclid_thumb_3 = np.sqrt((thumb_3.x - ref_0.x)**2 + (thumb_3.y -
ref_0.y)**2 + (thumb_3.z - ref_0.z)**2)
    thumb_3_list.append(euclid_thumb_3)
    euclid_thumb_4 = np.sqrt((thumb_4.x - ref_0.x)**2 + (thumb_4.y -
ref_0.y)**2 + (thumb_4.z - ref_0.z)**2)
    thumb_4_list.append(euclid_thumb_4)
    # euclidean index fing
    euclid_index_6 = np.sqrt((index_6.x - ref_0.x)**2 + (index_6.y -
ref_0.y)**2 + (index_6.z - ref_0.z)**2)
    index_6_list.append(euclid_index_6)
    euclid_index_8 = np.sqrt((index_8.x - ref_0.x)**2 + (index_8.y -
ref_0.y)**2 + (index_8.z - ref_0.z)**2)
    index_8_list.append(euclid_index_8)
    # euclidean mid fing
    euclid_mid_10 = np.sqrt((mid_10.x - ref_0.x)**2 + (mid_10.y - ref_0.y)**2
+ (mid_10.z - ref_0.z)**2)
    mid_10_list.append(euclid_mid_10)
    euclid_mid_12 = np.sqrt((mid_12.x - ref_0.x)**2 + (mid_12.y - ref_0.y)**2
+ (mid_12.z - ref_0.z)**2)
    mid_12_list.append(euclid_mid_12)
    # euclidean ring fing
```

```
    euclid_ring_14 = np.sqrt((ring_14.x - ref_0.x)**2 + (ring_14.y -
ref_0.y)**2 + (ring_14.z - ref_0.z)**2)
    ring_14_list.append(euclid_ring_14)
    euclid_ring_16 = np.sqrt((ring_16.x - ref_0.x)**2 + (ring_16.y -
ref_0.y)**2 + (ring_16.z - ref_0.z)**2)
    ring_16_list.append(euclid_ring_16)
    # euclidean pinky fing
    euclid_pinky_18 = np.sqrt((pinky_18.x - ref_0.x)**2 + (pinky_18.y -
ref_0.y)**2 + (pinky_18.z - ref_0.z)**2)
    pinky_18_list.append(euclid_pinky_18)
    euclid_pinky_20 = np.sqrt((pinky_20.x - ref_0.x)**2 + (pinky_20.y -
ref_0.y)**2 + (pinky_20.z - ref_0.z)**2)
    pinky_20_list.append(euclid_pinky_20)
    # concatenate
    rh_euclid = np.concatenate([thumb_3_list, thumb_4_list, index_6_list,
index_8_list, mid_10_list, mid_12_list,
                                ring_14_list, ring_16_list,
pinky_18_list,pinky_20_list])

    return rh_euclid
def left_pose_euclid(ref_11, elbow_13, wrist_15, pinky_17, index_19,
thumb_21):
    elbow_13_list = []
    wrist_15_list = []
    pinky_17_list = []
    index_19_list = []
    thumb_21_list = []
    # euclidean elbow
    euclid_elbow_13 = np.sqrt((elbow_13.x - ref_11.x)**2 + (elbow_13.y -
ref_11.y)**2 + (elbow_13.z - ref_11.z)**2)
    elbow_13_list.append(euclid_elbow_13)

    # euclidean wrist
    euclid_wrist_15 = np.sqrt((wrist_15.x - ref_11.x)**2 + (wrist_15.y -
ref_11.y)**2 + (wrist_15.z - ref_11.z)**2)
    wrist_15_list.append(euclid_wrist_15)

    # euclidean pinky
    euclid_pinky_17 = np.sqrt((pinky_17.x - ref_11.x)**2 + (pinky_17.y -
ref_11.y)**2 + (pinky_17.z - ref_11.z)**2)
    pinky_17_list.append(euclid_pinky_17)

    # euclidean index
    euclid_index_19 = np.sqrt((index_19.x - ref_11.x)**2 + (index_19.y -
ref_11.y)**2 + (index_19.z - ref_11.z)**2)
    index_19_list.append(euclid_index_19)

    # euclidean thumb
```

```
    euclidean_thumb_21 = np.sqrt((thumb_21.x - ref_11.x)**2 + (thumb_21.y -
ref_11.y)**2 + (thumb_21.z - ref_11.z)**2)
    thumb_21_list.append(euclidean_thumb_21)

    # concatenate
    l_pose_euclid = np.concatenate([elbow_13_list, wrist_15_list,
pinky_17_list, index_19_list, thumb_21_list])

    return l_pose_euclid
def right_pose_euclid(ref_12, elbow_14, wrist_16, pinky_18, index_20,
thumb_22):
    elbow_14_list = []
    wrist_16_list = []
    pinky_18_list = []
    index_20_list = []
    thumb_22_list = []
    # euclidean elbow
    euclid_elbow_14 = np.sqrt((elbow_14.x - ref_12.x)**2 + (elbow_14.y -
ref_12.y)**2 + (elbow_14.z - ref_12.z)**2)
    elbow_14_list.append(euclid_elbow_14)

    # euclidean wrist
    euclid_wrist_16 = np.sqrt((wrist_16.x - ref_12.x)**2 + (wrist_16.y -
ref_12.y)**2 + (wrist_16.z - ref_12.z)**2)
    wrist_16_list.append(euclid_wrist_16)

    # euclidean pinky
    euclid_pinky_18 = np.sqrt((pinky_18.x - ref_12.x)**2 + (pinky_18.y -
ref_12.y)**2 + (pinky_18.z - ref_12.z)**2)
    pinky_18_list.append(euclid_pinky_18)

    # euclidean index
    euclid_index_20 = np.sqrt((index_20.x - ref_12.x)**2 + (index_20.y -
ref_12.y)**2 + (index_20.z - ref_12.z)**2)
    index_20_list.append(euclid_index_20)

    # euclidean thumb
    euclidean_thumb_22 = np.sqrt((thumb_22.x - ref_12.x)**2 + (thumb_22.y -
ref_12.y)**2 + (thumb_22.z - ref_12.z)**2)
    thumb_22_list.append(euclidean_thumb_22)

    # concatenate
    r_pose_euclid = np.concatenate([elbow_14_list, wrist_16_list,
pinky_18_list, index_20_list, thumb_22_list])

    return r_pose_euclid
```

```
# Setup Folders for Collection
# Path for exported data, numpy arrays
DATA_PATH = os.path.join('newdata')

# Actions that we try to detect
actions = np.array(['Angry', 'Child', 'Come', 'Cry', 'Dad', 'Diligent',
'Dislike',
                    'Do', "Don't understand", 'Easy', 'Eat', 'Forget', 'Full',
'Go',
                    'Happy', 'Hard', 'Hi', 'How', 'Hungry', 'I', 'I am fine',
'Laugh',
                    'Lazy', 'Like', 'Love', 'Man', 'Miss', 'Mom', 'Name',
'Remember',
                    'Sad', 'Scare', 'Smile', 'Sorry', 'Thank you',
                    'Thank you very much', 'They', 'Today', 'Tomorrow',
'Understand',
                    'Want', 'We', 'What', 'When', 'Where', 'Why', 'Woman',
'Wonder',
                    'Work', 'Yesterday', 'You'])

# Thirty videos worth of data
no_sequences = 40

# Videos are going to be 60 frames in length
sequence_length = 60

for action in actions:
    for sequence in range(no_sequences):
        try:
            os.makedirs(os.path.join(DATA_PATH, action, str(sequence+40)))
        except:
            pass
```

```
# Collect Keypoint Values for Training and Testing
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:

    k = 8  # define the index of the action we want to collect

    for sequence in range(no_sequences):
        cap = cv2.VideoCapture(0)

        # Loop through video length aka sequence length
        for frame_num in range(sequence_length):
```

```
            # Read feed
            ret, frame = cap.read()

            # Make detections
            image, results = mediapipe_detection(frame, holistic)
            #print(results)

            # Draw landmarks
            draw_styled_landmarks(image, results)
            leftHandLandmark = results.left_hand_landmarks
            rightHandLandmark = results.right_hand_landmarks
            poseLandmark = results.pose_landmarks

            # Hand
            if leftHandLandmark:

                # left hand euclidean calculation
                # wrist
                lh_wrist_0 = leftHandLandmark.landmark[0]
                # thumb tip
                lh_thumb_3 = leftHandLandmark.landmark[3]
                lh_thumb_4 = leftHandLandmark.landmark[4]
                # index fing tip
                lh_ift_6 = leftHandLandmark.landmark[6]
                lh_ift_8 = leftHandLandmark.landmark[8]
                # mid fing tip
                lh_mft_10 = leftHandLandmark.landmark[10]
                lh_mft_12 = leftHandLandmark.landmark[12]
                # ring fing tip
                lh_rft_14 = leftHandLandmark.landmark[14]
                lh_rft_16 = leftHandLandmark.landmark[16]
                # pinky tip
                lh_pinky_18 = leftHandLandmark.landmark[18]
                lh_pinky_20 = leftHandLandmark.landmark[20]
                # euclidean distance
                left_hand_euclid = lh_euclid(lh_wrist_0, lh_thumb_3,
lh_thumb_4, lh_ift_6, lh_ift_8, lh_mft_10,
                                             lh_mft_12, lh_rft_14,
lh_rft_16, lh_pinky_18, lh_pinky_20)
            else:
                left_hand_euclid = np.zeros(10)

            if rightHandLandmark:

                # right hand euclidean calculation
                # wrist
                rh_wrist_0 = rightHandLandmark.landmark[0]
```

```
                # thumb tip
                rh_thumb_3 = rightHandLandmark.landmark[3]
                rh_thumb_4 = rightHandLandmark.landmark[4]
                # index fing tip
                rh_ift_6 = rightHandLandmark.landmark[6]
                rh_ift_8 = rightHandLandmark.landmark[8]
                # mid fing tip
                rh_mft_10 = rightHandLandmark.landmark[10]
                rh_mft_12 = rightHandLandmark.landmark[12]
                # ring fing tip
                rh_rft_14 = rightHandLandmark.landmark[14]
                rh_rft_16 = rightHandLandmark.landmark[16]
                # pinky tip
                rh_pinky_18 = rightHandLandmark.landmark[18]
                rh_pinky_20 = rightHandLandmark.landmark[20]
                # euclidean distance
                right_hand_euclid = rh_euclid(rh_wrist_0, rh_thumb_3,
rh_thumb_4, rh_ift_6, rh_ift_8, rh_mft_10,
                                              rh_mft_12, rh_rft_14,
rh_rft_16, rh_pinky_18, rh_pinky_20)
            else:
                right_hand_euclid = np.zeros(10)

            # Pose
            if poseLandmark:
                # pose euclid calculation
                left_shoulder_11 = poseLandmark.landmark[11]
                left_elbow_13 = poseLandmark.landmark[13]
                left_wrist_15 = poseLandmark.landmark[15]
                left_pinky_17 = poseLandmark.landmark[17]
                left_index_19 = poseLandmark.landmark[19]
                left_thumb_21 = poseLandmark.landmark[21]
                right_shoulder_12 = poseLandmark.landmark[12]
                right_elbow_14 = poseLandmark.landmark[14]
                right_wrist_16 = poseLandmark.landmark[16]
                right_pinky_18 = poseLandmark.landmark[18]
                right_index_20 = poseLandmark.landmark[20]
                right_thumb_22 = poseLandmark.landmark[22]

                # euclidean distance
                l_pose_euclid = left_pose_euclid(left_shoulder_11,
left_elbow_13, left_wrist_15,left_pinky_17,left_index_19,left_thumb_21)
                r_pose_euclid = right_pose_euclid(right_shoulder_12,
right_elbow_14, right_wrist_16,right_pinky_18,right_index_20,right_thumb_22)
            else:
                l_pose_euclid = np.zeros(5)
                r_pose_euclid = np.zeros(5)
```

```
            # NEW Apply wait logic
            if frame_num == 0:
                cv2.putText(image, 'STARTING', (120,200),
                            cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255, 0), 4,
cv2.LINE_AA)
                cv2.putText(image, 'Collecting frames for {} Video Number
{}'.format(actions[k], sequence+40), (15,12),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2,
cv2.LINE_AA)
                # Show to screen
                cv2.imshow('OpenCV Feed', image)
                cv2.waitKey(500)

            else:
                cv2.putText(image, 'Collecting frames for {} Video Number
{}'.format(actions[k], sequence+40), (15,12),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2,
cv2.LINE_AA)
                # Show to screen
                cv2.imshow('OpenCV Feed', image)

            keypoints = np.concatenate([left_hand_euclid, right_hand_euclid,
l_pose_euclid, r_pose_euclid])
            npy_path = os.path.join(DATA_PATH, actions[k], str(sequence+40),
str(frame_num))
            np.save(npy_path, keypoints)

            # Break gracefully
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

    cap.release()
    cv2.destroyAllWindows()
```

```
# Preprocess Data and Create Labels and Features
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
label_map = {label:num for num, label in enumerate(actions)}
sequences, labels = [], []
for action in actions:
    for sequence in np.array(os.listdir(os.path.join(DATA_PATH,
action))).astype(int):
        window = []
        for frame_num in range(sequence_length):
            res = np.load(os.path.join(DATA_PATH, action, str(sequence),
"{}.npy".format(frame_num)))
            window.append(res)
        sequences.append(window)
```

```
        labels.append(label_map[action])
```

```
# Create and train model
X = np.array(sequences)
y = to_categorical(labels).astype(int)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=101)

# import denpendencies tool
from keras.models import Model
from keras.layers import LSTM, Bidirectional, Dense, Conv2D, MaxPooling2D,
Flatten, Input, Reshape, Dropout, MultiHeadAttention, GlobalAveragePooling1D,
Activation, BatchNormalization, LayerNormalization
from keras.optimizers import Adam
from keras.callbacks import EarlyStopping, CSVLogger
from keras import Sequential
from keras.models import load_model
input_layer = Input(shape=(60, 30, 1))

reshape = Reshape((60, 30), input_shape=(60, 30, 1))(input_layer)
# print(reshape.shape)

lstm = LSTM(80, return_sequences=True)(reshape)
norm = LayerNormalization()(lstm)
actv = Activation(activation='relu')(norm)
dropout = Dropout(0.1)(actv)
# print(lstm.shape)

mul = MultiHeadAttention(num_heads=30, key_dim=5)
output_mul = mul(dropout, dropout)
# print(output_mul.shape)
norm = LayerNormalization()(output_mul)
ff = GlobalAveragePooling1D()(norm)
# print(ff.shape)

hidden = Dense(128, activation='relu')(ff)
hidden = Dense(100, activation='relu')(hidden)
hidden = Dense(64, activation='relu')(hidden)

output_layer = Dense(51, activation='softmax')(hidden)

model = Model(inputs=input_layer, outputs=output_layer)
model.summary()

model.compile(optimizer=Adam(),
              loss="categorical_crossentropy",
              metrics=['accuracy'])
callback = EarlyStopping(monitor='val_loss', patience=10, mode='min')
```

```
csv_logger = CSVLogger('log.csv', append=False, separator=';')

history = model.fit(x=x_train,
                    y=y_train,
                    batch_size=80,
                    epochs=2000,
                    verbose=1,
                    validation_data=(x_test, y_test),
                    callbacks=[callback, csv_logger])
```

```
# **Evaluate model**
from sklearn.metrics import accuracy_score, classification_report

model_pred = model.predict(x_test)
model_pred = np.argmax(model_pred, axis=1)

y_test = np.argmax(y_test, axis=1)
scores = accuracy_score(y_test, model_pred)
print("Scores: {}".format(scores))

report = classification_report(y_test, model_pred, output_dict=True)
pd.set_option('display.max_columns', None)
report_df = pd.DataFrame(report)
column_map = {"{}".format(num):name for num, name in enumerate(actions)}
report_df = report_df.rename(columns=column_map)
report_df.to_csv("report-97-79.csv")
report_df

# Collecting thai word
# create an empty dictionary to store the images
th_word = {}

# loop over the image files
for i in range(0, 50):
    # use the f-string to format the file name
    filename = 'Thai_word_edit/{}.jpg'.format(i)
    # read the image file using cv2.imread() function
    img = cv2.imread(filename)
    # resize the image
    img = cv2.resize(img, (img.shape[1]-50,40))
    # create a variable name for the image
    var_name = f"image_{i}"
    # store the image in the dictionary with its variable name
    th_word[var_name] = img


# Run program in real time
sequence_length = 60
```

```
flow_thres = 8
gesture_thres = 0.95
scale = 20
sequence = []
time = []
mean_mag = []
mean_magnitude = 0
res_lst = []
res_arr = 0
act = None
words = []
sentence = []
thsentence = []

cap = cv2.VideoCapture(0)
imgb = cv2.imread('Interface/b.jpg')
imgb_resize = cv2.resize(imgb,(1280,800))
imgborder = cv2.resize(imgb,(660,500))
imgline = cv2.resize(imgb,(1280,140))

kmitl = cv2.imread('Interface/KMITL.jpg')
kmitl_resize = cv2.resize(kmitl,(220,120))

imgb_resize[90:590,310:970] = imgborder
imgb_resize[600:740,0:1280] = imgline
imgb_resize[20:140,40:260] = kmitl_resize

cv2.putText(imgb_resize, 'Hand Sign Detection', (490,60),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2, cv2.LINE_AA)

# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        # Read feed
        ret, frame = cap.read()


        # Mediapipe
        image, results = mediapipe_detection(frame, holistic)

        # Draw landmarks
        draw_styled_landmarks(image, results)

        # 2. Prediction logic
        leftHandLandmark = results.left_hand_landmarks
        rightHandLandmark = results.right_hand_landmarks
        poseLandmark = results.pose_landmarks
```

```
        # Hand
        if leftHandLandmark:
            # left hand euclidean calculation
            # wrist
            lh_wrist_0 = leftHandLandmark.landmark[0]
            # thumb tip
            lh_thumb_3 = leftHandLandmark.landmark[3]
            lh_thumb_4 = leftHandLandmark.landmark[4]
            # index fing tip
            lh_ift_6 = leftHandLandmark.landmark[6]
            lh_ift_8 = leftHandLandmark.landmark[8]
            # mid fing tip
            lh_mft_10 = leftHandLandmark.landmark[10]
            lh_mft_12 = leftHandLandmark.landmark[12]
            # ring fing tip
            lh_rft_14 = leftHandLandmark.landmark[14]
            lh_rft_16 = leftHandLandmark.landmark[16]
            # pinky tip
            lh_pinky_18 = leftHandLandmark.landmark[18]
            lh_pinky_20 = leftHandLandmark.landmark[20]
            # euclidean distance
            left_hand_euclid = lh_euclid(lh_wrist_0, lh_thumb_3, lh_thumb_4,
lh_ift_6, lh_ift_8, lh_mft_10,
                                         lh_mft_12, lh_rft_14, lh_rft_16,
lh_pinky_18, lh_pinky_20)
        else:
            left_hand_euclid = np.zeros(10)
        if rightHandLandmark:
                # right hand euclidean calculation
                # wrist
                rh_wrist_0 = rightHandLandmark.landmark[0]
                # thumb tip
                rh_thumb_3 = rightHandLandmark.landmark[3]
                rh_thumb_4 = rightHandLandmark.landmark[4]
                # index fing tip
                rh_ift_6 = rightHandLandmark.landmark[6]
                rh_ift_8 = rightHandLandmark.landmark[8]
                # mid fing tip
                rh_mft_10 = rightHandLandmark.landmark[10]
                rh_mft_12 = rightHandLandmark.landmark[12]
                # ring fing tip
                rh_rft_14 = rightHandLandmark.landmark[14]
                rh_rft_16 = rightHandLandmark.landmark[16]
                # pinky tip
                rh_pinky_18 = rightHandLandmark.landmark[18]
                rh_pinky_20 = rightHandLandmark.landmark[20]
                # euclidean distance
```

```
                right_hand_euclid = rh_euclid(rh_wrist_0, rh_thumb_3,
rh_thumb_4, rh_ift_6, rh_ift_8, rh_mft_10,
                                              rh_mft_12, rh_rft_14,
rh_rft_16, rh_pinky_18, rh_pinky_20)
        else:
            right_hand_euclid = np.zeros(10)
        # Pose
        if poseLandmark:
            # pose euclid calculation
            left_shoulder_11 = poseLandmark.landmark[11]
            left_elbow_13 = poseLandmark.landmark[13]
            left_wrist_15 = poseLandmark.landmark[15]
            left_pinky_17 = poseLandmark.landmark[17]
            left_index_19 = poseLandmark.landmark[19]
            left_thumb_21 = poseLandmark.landmark[21]
            right_shoulder_12 = poseLandmark.landmark[12]
            right_elbow_14 = poseLandmark.landmark[14]
            right_wrist_16 = poseLandmark.landmark[16]
            right_pinky_18 = poseLandmark.landmark[18]
            right_index_20 = poseLandmark.landmark[20]
            right_thumb_22 = poseLandmark.landmark[22]
            # euclidean distance
            l_pose_euclid = left_pose_euclid(left_shoulder_11, left_elbow_13,
left_wrist_15, left_pinky_17, left_index_19, left_thumb_21)
            r_pose_euclid = right_pose_euclid(right_shoulder_12,
right_elbow_14, right_wrist_16, right_pinky_18, right_index_20,
right_thumb_22)
        else:
            l_pose_euclid = np.zeros(5)
            r_pose_euclid = np.zeros(5)
        keypoints =
np.concatenate([left_hand_euclid,right_hand_euclid,l_pose_euclid,r_pose_euclid
])
        sequence.append(keypoints)

        image_resize = cv2.resize(image, (640,480))

        imgb_resize[100:580,320:960] = image_resize

        # Operation

        if len(sequence) == 1:
            cv2.circle(imgb_resize, (640,130), 10, (0, 255, 0), -1)
        if len(sequence) == 60:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            print('LSTM [{}] : Softmax [{}]'.format(actions[np.argmax(res)],
np.max(res)))
            if np.max(res) > gesture_thres:
```

```
                sentence.append(actions[np.argmax(res)])
                thsentence.append((th_word['image_{}'.format(np.argmax(res))])
)

            sequence = []
            print(sentence)
            print(len(thsentence))


        if np.max(res) > gesture_thres:
            cv2.rectangle(imgb_resize, (320,110), (460,150), (0,0,0) , -1)
            cv2.putText(imgb_resize, '{}'.format(actions[np.argmax(res)]),
(330,140), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2, cv2.LINE_AA)


        if np.max(res) < gesture_thres:
            cv2.rectangle(imgb_resize, (320,110), (460,150), (0,0,0) , -1)
            cv2.putText(imgb_resize, 'None', (330,140),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2, cv2.LINE_AA)
            # cv2.putText(image, '{}'.format(mean_magnitude), (3,30),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2, cv2.LINE_AA)

        if sentence.count('They') > 1:
            cv2.putText(imgb_resize, 'Found "They" reduplication', (440,340),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 3, cv2.LINE_AA)
            cv2.imshow('Frame', imgb_resize)
            cv2.waitKey(2000)
            sentence = []
            thsentence = []

        if sentence.count('Smile') > 1:
            cv2.putText(imgb_resize, 'Found "Smile" reduplication', (440,340),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 3, cv2.LINE_AA)
            cv2.imshow('Frame', imgb_resize)
            cv2.waitKey(2000)
            sentence = []
            thsentence = []

        if len(sentence) > 5:
            sentence = []
            thsentence = []
            cv2.rectangle(imgb_resize, (870,120), (955,160), (0,0,0) , -1)
            cv2.putText(imgb_resize, 'Clear',
(880,150),cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv2.LINE_AA)
            cv2.imshow('Frame', imgb_resize)
            cv2.waitKey(2000)
        # cv2.rectangle(imgb_resize, (200,580), (840,630),(0,0,0) , -1)
        cv2.rectangle(imgb_resize, (870,120), (955,160),(0,0,0) , -1)
```

```
        cv2.putText(imgb_resize, 'Start', (880,150),cv2.FONT_HERSHEY_SIMPLEX,
0.8, (0, 255, 0), 2, cv2.LINE_AA)

        if cv2.waitKey(10) & 0xFF == ord('s'): # push s to send message

            for i in range(len(sentence)):

                if i == 0:
                    r = 330+(thsentence[i].shape[1])
                    imgb_resize[660:700,330:r] = thsentence[i]

                elif i != 0:

                    imgb_resize[660:700,r:r+(thsentence[i].shape[1])] =
thsentence[i]
                    r = r+(thsentence[i].shape[1])

            cv2.putText(imgb_resize, ' '.join(sentence),
(330,630),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2, cv2.LINE_AA)
            cv2.rectangle(imgb_resize, (870,120), (955,160), (0,0,0) , -1)
            cv2.putText(imgb_resize, 'Wait',
(880,150),cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 0), 2, cv2.LINE_AA)
            cv2.imshow('Frame', imgb_resize)
            cv2.waitKey(5000)
            sentence = []
            thsentence = []
            imgb_resize[600:740,0:1280] = imgline

        if cv2.waitKey(10) & 0xFF == ord('d'): # push d to clear message

            sentence = []
            thsentence = []
            cv2.rectangle(imgb_resize, (870,120), (955,160), (0,0,0) , -1)
            cv2.putText(imgb_resize, 'Clear',
(880,150),cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv2.LINE_AA)
            cv2.imshow('Frame', imgb_resize)
            cv2.waitKey(2000)

        if cv2.waitKey(10) & 0xFF == ord('q'): # push q to end program
            break

        # Opens a new window and displays the output frame
        cv2.imshow("Frame", imgb_resize)

cap.release()
cv2.destroyAllWindows()
```