บทความวิจัย

แปลภาษามือด้วยการเรียนรู้เชิงลึก

Sign Language Translation Using Deep Learning

นิติพล ขจรภพ, อภิสิทธิ์ กิจเจริญปัญญา, พรทิวา ปะวะระ
สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาการสารสนเทศ มหาวิทยาลัยมหาสารคาม
nitiponkhachornphop@gmail.com, Lego1851@gmail.com, porntiwa.p@msu.ac.th

บทคัดย่อ

โครงการนี้เป็น Desktop Application ที่ใช้เทคโนโลยี LSTM เพื่อแปลภาษามือเป็นข้อความที่สามารถเข้าใจได้โดยมุ่งเน้นช่วยผู้ที่มีความบกพร่องทางร่างกายหรือบกพร่องการได้ยินและการสื่อสาร. โครงการนี้เสนอแนวทางในการใช้เทคโนโลยีเพื่อเพิ่มโอกาสและคุณภาพชีวิตของกลุ่มคนในสังคม.

1. บทนำ

ในปัจจุบันการใช้ภาษามือเพื่อการสื่อสารได้ถูกนำมาใช้โดยกลุ่มคนที่มีปัญหาทางด้านร่างกายเช่น ผู้ที่บกพร่องทางการได้ยินหรือผู้ที่บกพร่องทางการสื่อสารได้ ซึ่งการใช้ภาษามือนั้นมีส่วนสำคัญอยู่หลายส่วนเช่น การใช้ท่าทาง การใช้มือ หรือบางประเภทต้องใช้สีหน้าในการบอกอารมณ์และความรู้สึกด้วย จึงเป็นการยากที่ผู้ที่ไม่ได้ฝึกภาษามือจะทำการเข้าใจ

ด้วยเหตุผลดังกล่าว จึงทำให้การที่ผู้ที่ไม่ได้ฝึกภาษามือไม่เข้าใจว่าการใช้ท่าทางเหล่านี้แปลว่าอะไร แต่ในปัจจุบัน มีเทคโนโลยีที่ตอบสนองความต้องการของมนุษย์มากขึ้น ทำให้เราสามารถใช้ชีวิตได้สะดวกสบายมากยิ่งขึ้น เทคโนโลยีทางด้านปัญญาประดิษฐ์หรือที่

เรียกว่า Artificial Intelligence (AI) ได้ถูกนำมาประยุกต์ใช้ในสังคมได้อย่างแพร่หลาย เช่น ใช้ในการตรวจจับท่าทางภาษามือเพื่อใช้ในการสื่อสาร และเพื่อที่จะเป็นประโยชน์ต่อกลุ่มผู้คนเหล่านี้ในการสื่อสารกับผู้คนปกติทั่วไปให้ได้ง่ายยิ่งขึ้น

ดังนั้น ผู้จัดทำจึงขอนำเสนอโปรแกรมในการแปลภาษามือ ซึ่งเป็นเดสก์ท็อปแอปพลิเคชัน ที่ผู้ใช้สามารถทำความเข้าใจภาษามือได้โดยง่าย โดยให้ผู้ใช้งานแสดงท่าทางภาษามือผ่านหน้ากล้อง จากนั้นเว็บไซต์จะทำการเปรียบเทียบกับโมเดลการตรวจจับท่าทางภาษามือ ด้วยอัลกอริทึม LSTM(Long Short-Term Memory) ว่าท่าทางเหล่านั้นมีความหมายว่าอะไร ด้วยเว็บไซต์นี้ไม่ใช่แค่จะทำให้ผู้ที่ไม่ได้ฝึกฝนภาษามือสามารถเข้าใจภาษามือได้ง่ายขึ้นเท่านั้น แต่ยังทำให้กลุ่มคนที่มีปัญหาทางด้านร่างกาย เช่น ผู้ที่บกพร่องทางการได้ยินหรือผู้ที่บกพร่องทางการสื่อสารนั้น สามารถสื่อสารกับผู้ที่ไม่ได้ฝึกฝนภาษามือได้ง่ายขึ้นอีกด้วย

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

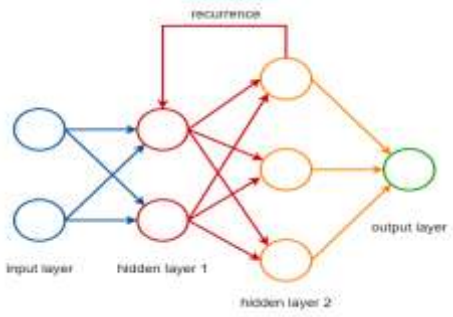
2.1 ทฤษฎีที่เกี่ยวข้อง

การทำงานของการเรียนรู้เชิงลึกนั้นเป็นการทำงานที่เลียนแบบมาจากการทำงานของระบบโครงข่ายประสาทในสมองของมนุษย์ โดยอัลกอริทึมของกาเรียนรู้เชิงลึกถูกสร้างจากการนำเอาNeural network หลายๆชั้น (Layer) มาซ้อน (Stack) ต่อกัน โดยชั้นแรกสุดจะทำหน้าที่ในการรับข้อมูล (Input layer) ชั้นสุดท้ายจะทำหน้าที่ส่งผลลัพธ์การประมวลผลออกมา (Output layer) ส่วนชั้นระหว่างชั้นแรกสุดกับชั้นสุดท้าย จะถูกเรียกว่าชั้นซ่อน (Hidden layer)

2.1.1.1 Recurrent Neural Networks (RNN) [2] [3] [5]

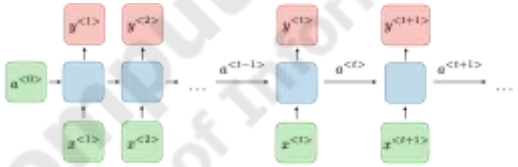
Recurrent Neural Networks หรือ

โครงข่ายแบบวนซ้ำ คือNeural networks หลายชั้นที่สามารถเก็บข้อมูลไว้ที่โหนดจึงทำให้สามารถเก็บข้อมูลแบบลำดับและแสดงผลลัพธ์ออกมาเป็นลำดับซ้ำยังสามารถต่อกันเป็นวงวน (Loop) ได้ เพราะฉะนั้น RNN จึงเหมาะกับการประมวลผลกับข้อมูลที่เป็นลำดับ อย่าง วิดีโอเป็นอย่างยิ่ง



ภาพประกอบที่ 2.1 การทำงานของโครงข่ายประสาทเทียมแบบ RNN

RNN เป็นหนึ่งการเรียนรู้เชิงลึก ที่จะอนุญาตให้ ข้อมูลส่งออก ตัวก่อนหน้า สามารถใช้เป็น ข้อมูลนำเข้า ใน state ต่อไปของชั้นซ่อน (Hidden state) ได้ โดย RNN จะมีโครงสร้างตาม ภาพประกอบที่ 2.2 โครงสร้างของ RNN ดังนี้



ภาพประกอบที่ 2.2 โครงสร้างของ RNN [4]

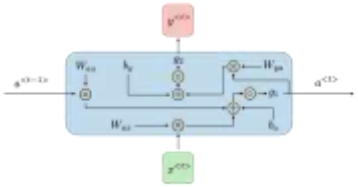
โดย

t คือ จำนวนครั้งในแต่ละรอบ

a<t> คือ ผลลัพธ์ที่ได้จากการคำนวณในรอบก่อนหน้า

x<t> คือ ข้อมูลนำเข้า ณ โหนดนั้นๆ ในแต่ละรอบ

*y<t>*คือ ข้อมูลส่งออก ณ โหนดนั้นๆ ในแต่ละรอบ

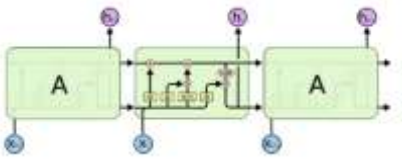


ภาพประกอบที่ 2.3 โครงสร้างในชั้นซ่อนของ RNN [4]

2.1.2 Long Short Term Memory (LSTM) [8]

Long Short Term Memory หรือ LSTM เป็นหนึ่งในสับเซ็ตย่อยของ Recurrent Neural Network หรือ RNN ซึ่งเป็นโครงข่ายประสาทเทียมแบบวนซ้ำ ที่มีความสามารถในการเรียนรู้ข้อมูลที่เป็น long-term และมีลำดับได้ โดย LSTM ได้ถูกพัฒนาขึ้นมาเพื่อจัดการกับข้อจำกัดของ RNN ที่มีปัญหา Vanishing gradient ซึ่งจะเกิดขึ้นเมื่อตอนที่ทำการเทรนโมเดลแล้วมีการอัพเดทพารามิเตอร์ซ้ำๆ ในกระบวนการ Gradient descent หรือกระบวนการที่จะทำการอัพเดทค่า Weight ในโมเดลเพื่อทำให้ Cost function ลดลงไปที่จุดต่ำสุด ด้วยกระบวนการนี้ จะทำให้ค่า Weight บางตัวมีค่าน้อยลงจนเหลือน้อยมากๆ โดยเฉพาะตัวที่อยู่ใน Layer ชั้นแรกๆ ที่ใกล้กับ Input layer ส่งผลให้การอัพเดต พารามิเตอร์ครั้งถัดไป Loss จะเปลี่ยนแปลงน้อยมากจนไม่เกิดผลในการ Optimise หรืออาจจะต้องใช้เวลาและจำนวนรอบสูงมากๆในการเทรน

โครงสร้างของ LSTM จะมีลักษณะคล้ายโซ่ โดยในแต่ละเซลล์จะมีส่วนประกอบดังรูป



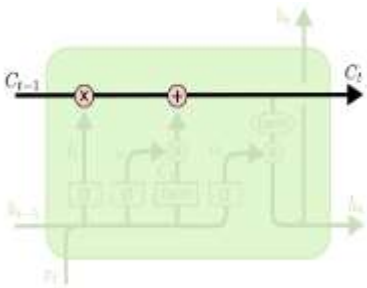
ภาพประกอบที่ 2.4 โครงสร้างของ LSTM [8]

LSTM จะมีส่วนประกอบอยู่ 4 ส่วน ได้แก่ Cell state, Forget gate, Input gate

และ Output gate เพื่อควบคุมการไหลผ่านของข้อมูล โดยแต่ละส่วนจะสามารถอธิบายได้ดังนี้

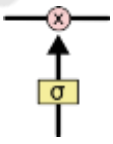
3.1.1.1 Cell state

คีย์สำคัญของ LSTM ก็คือ Cell state ซึ่งเป็นเส้นแนวนอนที่อยู่ด้านบนของเซลล์ ดังภาพประกอบ



ภาพประกอบที่ 2.5 Cell State [8]

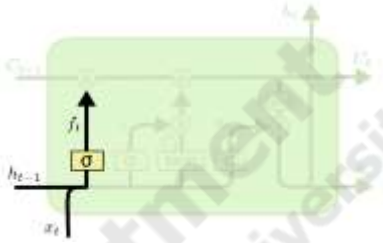
Cell state เป็นตัวเก็บ State ของ Memory cell ใน LSTM ซึ่ง Cell state เป็นเหมือนกับสายพานลำเลียง โดยมันจะวิ่งตรงผ่านไปทั้งเซลล์โดยแทบไม่ได้ยุ่งเกี่ยวกับ Gate ใดๆ ดังนั้นจึงเป็นเรื่องง่ายมากที่ข้อมูลที่ไหลผ่าน Cell state จะไม่มีการเปลี่ยนแปลง นอกจากนี้ LSTM ยังมีความสามารถในการลบหรือเพิ่มข้อมูลไปยัง Cell state ซึ่งจะถูกควบคุมโดยโครงสร้างที่เรียกว่า Gates โดย Gates เป็นตัวที่จะปล่อยให้ข้อมูลผ่านไปได้ ซึ่ง Gates ประกอบด้วย Sigmoid layer และ Pointwise multiplication operations



ภาพประกอบที่ 2.6 Gates [8]

2.1.2.2 Forget gate

ส่วนนี้จะเป็นส่วนที่รับผิดชอบในการตัดสินใจว่า Cell state ไหนที่จะเอาทิ้งไป หรือ Cell state ไหนที่จะเก็บไว้ โดยข้อมูลที่จะเอาทิ้งไปอาจจะเป็นข้อมูลบางส่วนที่ได้มาจากเซลล์ก่อนหน้า ซึ่ง Forget gate เป็นเหมือนตัวตัดสินหากให้ค่าเป็น 0 ก็จะลบ Cell state เดิมออก แต่ถ้าหากให้ค่าเป็น 1 ก็จะเก็บ Cell state นี้ต่อไป และการสร้าง Forget gate นี้ จะดูจากข้อมูลที่นำเข้ามา ประกอบกับเซลล์ก่อนหน้าประกอบการตัดสินใจ



ภาพประกอบที่ 2.7 Forget Gate [8]

โดยในการตัดสินใจว่าจะให้ Cell state ไหนผ่านไปได้นั้น จะใช้ Sigmoid activation function (ฟังก์ชันที่มีข้อมูลส่งออกเป็นตัวเลข 0 และ 1 เพื่อใช้อธิบายว่าควรจะปล่อยให้ Cell state ไหนผ่านไปได้ โดย 0 จะหมายถึง ไม่ให้ผ่านไป และ 1 จะหมายถึง ให้ผ่านได้) เป็นตัวตัดสินใจ ซึ่งสมการของ Sigmoid activation function มีดังนี้

$$f_t = \sigma(W_{xf}[h_{t-1}, x_t] + b_f)$$

โดย

$\mathbf{f_t}$ คือ ข้อมูลส่งออกที่เป็นเวกเตอร์

$\boldsymbol{\sigma}$ คือ Sigmoid activation function

$\mathbf{W_{xf}}$ คือ ค่า Weight matrix

$\mathbf{h_{t-1}}$ คือ State ก่อนหน้า

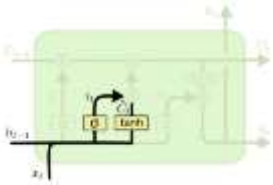
$\mathbf{x_t}$ คือ ข้อมูลนำเข้าใน State ปัจจุบัน

$\mathbf{b_f}$ คือ ค่า Bias term

2.1.2.3 Input Gate

ส่วนนี้จะเป็นส่วนที่จะเลือกว่าจะนำเอาข้อมูลที่เข้ามาใหม่ตัวไหนไปเก็บลงยัง Cell state โดยในส่วนนี้จะแบ่งส่วนย่อยออกอีก 2 ส่วน ส่วนแรกถูกเรียกว่า “input gate layer” ซึ่งเป็นตัวที่จะตัดสินใจเลือกว่าจะทำการอัพเดท Cell state ด้วย Input data ใหม่ดีหรือไม่ และหากจะอัพเดทจะอัพเดทด้วยค่า

อะไร ส่วนที่สองถูกเรียกว่า “tanh layer” ซึ่งจะทำหน้าที่สร้างเวกเตอร์ของ Candidate value ที่จะสามารถเพิ่มเข้าไปยัง Cell state ได้ จากนั้นจะรวมทั้งสองส่วนเข้าด้วยกันเพื่อทำการสร้างการอัพเดทใน Cell state นั้น



ภาพประกอบที่ 2.8 Input Gate [8]

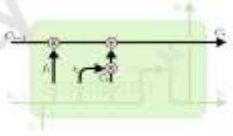
ในการจะอัพเดท Cell state ตัวเดิม (C_{t-1}) ให้เป็น Cell state ตัวใหม่ (C_t) ได้นั้น จะยังคงทำการใช้ Sigmoid function เป็นตัวตัดสินว่าจะอนุญาตให้อัพเดทหรือไม่ โดยจะใช้ค่าจาก Cell state ตัวก่อนหน้ากับค่า Input data มาคำนวณ โดยสมการมีดังนี้

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

จากนั้นในการจะอัพเดทจริงๆ จะใช้สิ่งที่เรียนกว่า Input modulation gate โดยสมการจะคล้ายกับ Input gate แต่ว่าจะใช้เป็น tanh function แทน โดยสมการมีดังนี้

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

จากสมการข้างต้นจะทำให้ได้ Candidate values ตัวใหม่ที่จะนำไปอัพเดท State value



ภาพประกอบที่ 2.9 Input Gate (ต่อ) [8]

ซึ่งหลังจากที่เราได้ข้อมูลจาก Forget gate, Input gate และ Input modulation gate แล้ว ก็จะเพียงพอต่อการนำไปทำการอัพเดท Cell state โดยใช้สมการดังต่อไปนี้

$$\mathrm{C_t = f_t * C_{t-1} + i_t * \tilde{C}_t}$$

จากสมการในส่วนทาง หาก Forget gate ต้องการลบ Cell state เดิมทิ้ง (f_t มีค่าเป็น 0) ก็จะไม่เอาค่า C_{t-1} มาทำการอัพเดท Cell state แต่ถ้าหาก f_t มีค่าเป็น 1 ก็จะคงค่า C_{t-1} ไว้เพื่อทำการอัพเดทต่อไป และในส่วนทางขวาของสมการ จะเป็นการอัพเดท Cell state จากข้อมูลใหม่ ซึ่งในขั้นตอนนี้จะได้ค่าที่จะทำการอัพเดทเตรียมพร้อมไว้แล้วจาก Input modulation gate จากนั้นจะทำการตัดสินใจว่า ค่าที่ Input modulation gate ได้เตรียมไว้จะนำมาใช้หรือไม่ โดยจะใช้ข้อมูลส่งออกจาก Input gate มาทำการตัดสินใจ หากข้อมูลจาก Input gate มีค่าเป็น 0 จะไม่นำข้อมูลมาใช้ แต่หากข้อมูลจาก Input gate มีค่าเป็น 1 จึงจะนำข้อมูลมาใช้ จากค่าทั้งหมดที่ได้มา จะทำให้ได้ค่า C_t ตัวใหม่

2.1.2.4 Output Gate

ส่วนนี้จะเป็นส่วนที่จะตัดสินใจว่าจะส่งข้อมูลอะไรออกไป โดยข้อมูลที่จะส่งออกไปจะเป็นข้อมูลที่ได้รับการกรองมาแล้ว ขั้นแรก ใช้ Sigmoid layer เพื่อตัดสินใจว่าจะเอาส่วนไหนของ Cell state ที่จะทำการส่งข้อมูลออกไป จากนั้นใส่ Cell state ผ่าน tanh (เพื่อให้ได้ค่าระหว่าง -1 และ 1) และจึงทำการคูณด้วยข้อมูลที่ได้รับมาจาก Sigmoid gate เพื่อให้ได้ส่งออกข้อมูลเฉพาะส่วนที่ได้เลือก
โดยสมการ มีดังนี้

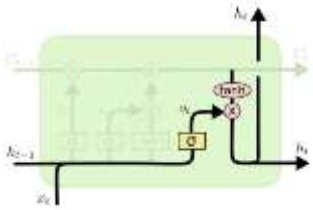
$$o_t = \sigma(W_o[h_{t-1,}\, x_t] + b_o)$$

หลังจากนั้น จะคำนวณหาค่า h_t เพื่อนำไปใช้สำหรับ Time step ถัดไป โดยสมการมีดังนี้

$$h_t = o_t \odot tanh(c_t)$$

หาก output gate ให้ตัว o_t มีค่าเป็น 0 แล้ว ค่าของ h_t ก็จะมีค่าเป็น 0 หรือก็คือ จะไม่ส่งค่าใดๆออกไป ในขณะเดียวกันหาก o_t มีค่าเป็น 1 ก็จะทำการคำนวณค่า h_t และส่งออก

ไปยัง time step ถัดไป โดยลักษณะของ Output gate สามารถดูได้ได้ดังภาพนี้



ภาพประกอบที่ 2.10 Output Gate [8]

2.1.3 Mediapipe [7]

Mediapipe เป็น Open source cross-platform framework ที่สร้างขึ้นโดย Google เพื่อใช้สร้าง Pipeline สำหรับการประมวลผลข้อมูลการรับรู้จากรูปแบบต่างๆ เช่น วิดีโอและเสียง เป็นต้น การใช้ Mediapipe นั้นสามารถนำไปประยุกต์ใช้ได้หลายอย่าง ไม่ว่าจะเป็นการตรวจจับการเคลื่อนไหวของมนุษย์ การจดจำใบหน้า และการจดจำท่าท่างภาษามือ เป็นต้น โดยในที่นี้จะใช้ Mediapipe Hands เพื่อทำการตรวจจับมือ Mediapipe Pose เพื่อทำการตรวจจับท่าทาง และ Mediapipe Face Mesh เพื่อทำการตรวจจับใบหน้า ซึ่งในการตรวจจับ จะได้ข้อมูล Key point ออกมา ซึ่งข้อมูลหลักๆจะประกอบด้วย x คือค่าความกว้าง y คือค่าความยาว และ z คือค่าความลึก โดยจะอธิบายการทำงานของ Mediapipe Hands, Mediapipe Pose และ Mediapipe Face Mesh ได้ดังต่อไปนี้

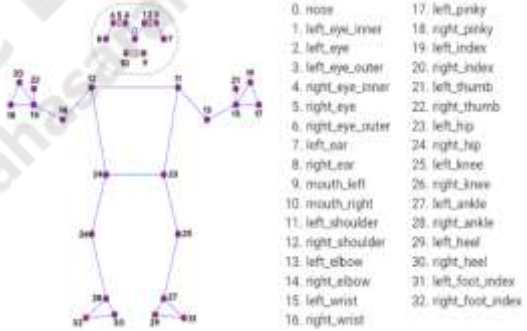
2.1.3.1 Mediapipe Hands

Mediapipe hands เป็นหนึ่งในโมดูลของ Mediapipe ซึ่งเป็นการตรวจจับหรือติดตามฝ่ามือและนิ้วมือที่มีความแม่นยำสูง โดยใช้การเรียนรู้ของเครื่องเพื่อหาจุดสังเกต (Landmarks) ของมือแบบ 3 มิติ ได้แก่ x, y และ z ซึ่งจะมีอยู่ 21 จุดในหนึ่งเฟรม ดังภาพ

ภาพประกอบที่ 2.11 Hand Landmark [7]

2.1.3.2 Mediapipe Pose

Mediapipe pose เป็นหนึ่งในโมดูลของ Mediapipe ซึ่งเป็นการตรวจจับหรือติดตามท่าทางร่างกายที่มีความแม่นยำสูง ซึ่งจะหาจุด Landmark ของท่าทางแบบ 3 มิติ ได้แก่ x, y, z และ visibility เพื่อมาใช้เป็นตัวบอกว่าตัวโมเดลได้มีการมองเห็นในส่วนของท่าทางหรือไม่ โดยจุด Landmarks มีทั้งหมด 33 จุด ดังรูป



ภาพประกอบที่ 2.12 Pose Landmark [7]

โดยในงานนี้เราใช้การตรวจจับท่าทางของร่างกายเพื่อให้รู้ว่า มืออยู่ส่วนไหนของเฟรม เพื่อให้การตรวจจับมือมีความแม่นยำมากยิ่งขึ้น

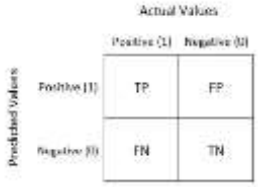
2.1.3.3 Mediapipe Face Mesh

Mediapipe Face Mesh เป็นหนึ่งในโมดูลของ Mediapipe ซึ่งเป็นการตรวจจับจุดสังเกตุหรือจุด Landmarks บนใบหน้าในรูปแบบ 3 มิติโดยใช้การเรียนรู้ของเครื่อง ซึ่งได้แก่ x, y และ z โดยมีจุด Landmarks อยู่ทั้งหมด 468 จุด

2.1.4 Dense [9]

Dense layer หรืออีกชื่อคือ Fully connected layer เป็นชั้นของเซลล์ประสาทอย่างง่าย ที่แต่ละเซลล์นั้นรับข้อมูลมาจากเซลล์

ประสาททั้งหมดในชั้นก่อนหน้า การทำงาน คือ เซลล์ประสาทของ Dense layer จะทำการคูณเมทริกซ์กับเวกเตอร์ โดยที่เวกเตอร์แถวของข้อมูลส่งออกจากชั้นก่อนหน้าเท่ากับเวกเตอร์คอลัมน์ของ Dense layer



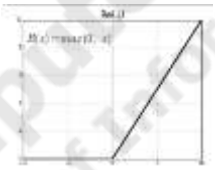
ภาพประกอบที่ 2.13 Dense Layers [9]

2.1.5 Activation function

Activation function ที่ใช้ในงานวิจัยนี้ มีทั้งหมด 2 ฟังก์ชัน ดังต่อไปนี้

2.1.5.1 ReLU Function [10]

ReLU นั้นจะทำให้ข้อมูลส่งออกมีค่าอยู่ระหว่าง 0 ถึง Infinity ถ้าค่าที่ Input เข้าไป ต่ำกว่า 0 จะถูกแปลงให้เป็น 0 ทั้งหมด แต่ถ้า Input มากกว่า 0 ก็จะเป็น Linear function คือไม่มีการ เปลี่ยนแปลง สำหรับ ReLU การไล่ระดับสีจะเป็นค่าบวกเสมอ ทำให้ลดการกระจัดกระจายและความ น่าจะเป็นที่จะเกิด Vanishing gradient



ภาพประกอบที่ 2.14 Rectified Liner Unit (ReLU) [10]

2.1.6 Confusion Matrix [11]

Confusion Matrix คือ การวัดประสิทธิภาพการทำนายผลของโมเดล โดย Confusion matrix จะมีไอเดียจากการวัดว่าสิ่งที่โมเดลทำนายกับสิ่งสิ่งที่เกิดขึ้นจริง มีสัดส่วนเป็นอย่างไรโดยสามารถดูได้ตามภาพประกอบ ดังต่อไปนี้

$$Ax = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix},$$

ภาพประกอบที่ 2.15 Confusion Matrix [11]

- TP (True Positive) ข้อมูลที่ทำนายตรงกับสิ่งที่เกิดขึ้นจริง ในกรณีที่ทำนายว่าเป็นจริงและสิ่งที่เกิดขึ้นเป็นจริง
- TN (True Negative) ข้อมูลที่ทำนายตรงกับสิ่งที่เกิดขึ้นจริง ในกรณีที่ทำนายว่าไม่เป็นจริง และสิ่งที่เกิดขึ้นไม่เป็นจริง
- FP (False Positive) ข้อมูลที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้นจริง คือทำนายว่าเป็นจริงแต่สิ่งที่เกิดขึ้นไม่เป็นจริง
- FN (False Negative) ข้อมูลที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้นจริง คือทำนายว่าไม่เป็นจริง แต่สิ่งที่เกิดขึ้นเป็นจริง

2.1.6.1 การคำนวณค่า Confusion Matrix

Accuracy คือ ค่าความถูกต้อง คำนวณด้วยการนำผลรวมของแส้นทแยงหารด้วยผลรวมของทุกค่าในตาราง โดยสมการมีดังนี้

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision คือ ค่าความแม่นยำ โดยจะแสดงความแม่นยำของคลาสที่เป็นการคาดการณ์ว่าเป็นผลบวกว่ามีมากน้อยเพียงใด โดยสมการมีดังนี้

$$precision = \frac{TP}{TP + FP}$$

Recall คือ ค่าระลึก โดยจะคำนวณอัตราส่วนของคลาสที่คาดการณ์เป็นผลบวกที่ตรวจพบวัตถุได้อย่างถูกต้อง โดยค่าระลึกนี้จะแสดงให้เห็นว่าตัวโมเดลนั้นสามารถรับรู้ถึงคลาสที่เป็นผลบวกได้ดีแค่ไหน โดยสมการมีดังนี้

$$recall = \frac{TP}{TP + FN}$$

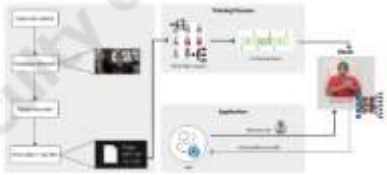
F1-score คือ เป็นการนำค่าที่ได้จาก precision และ recall มาคำนวณรวมกัน เพื่อหาค่า Harmonic mean โดยสมการมีดังนี้

$$F1 = \frac{precision \times recall}{precision + recall}$$

2.2 งานวิจัยที่เกี่ยวข้อง

ผลงานวิจัยของ Ankita Wadhawan และ Parteek Kumar เรื่อง “Deep learning-based sign language recognition system for static signs” [6] ที่ออกมาแก้ปัญหาการสื่อสารด้วยภาษามือระหว่างผู้ที่บกพร่องทางการได้ยินหรือผู้ที่บกพร่องทางการสื่อสารกับบุคคลทั่วไป โดยการใช้ Convolutional Neural Network (CNN) ในการสร้างโมเดล ซึ่งมีผลเฉลยทั้งหมด 100 ท่า ซึ่งโมเดลนี้ได้ประสิทธิภาพถึง 99.72 เปอร์เซ็นต์สำหรับภาพสี และ 99.90 เปอร์เซ็นต์ สำหรับภาพที่เป็นระดับสีเทา แต่โมเดลดังกล่าวนี้สามารถจำแนกได้เฉพาะท่าท่างที่เป็นท่างที่อยู่นิ่ง ไม่ได้มีการขยับ ซึ่งท่าทางภาษามือนั้นจำเป็นที่จะต้องมีการแสดงสีหน้าและท่าทางเพื่อที่จะสามารถทำให้ท่าทางเหล่านั่นมีความหมายได้

3. แผนการดำเนินงาน



ภาพประกอบที่ 3.1 แผนการดำเนินงาน

ในบทนี้จะกล่าวถึงขั้นตอนการดำเนินงานของโครงงานปริญญานิพนธ์ ซึ่งจะทำให้ทราบถึงการวิเคราะห์ระบบ กระบวนการประมวลผลของระบบ และขั้นตอนการทำงานของระบบเป็นอย่างไร โดยขั้นตอนในการดำเนินงานวิจัยมีรายละเอียดดังนี้

1. ขั้นตอนการจัดเตรียมข้อมูล
2. ขั้นตอนการพัฒนาโมเดล
3. ขั้นตอนการเรียนรู้และทดสอบโมเดล
4. การวัดประสิทธิภาพของโมเดล

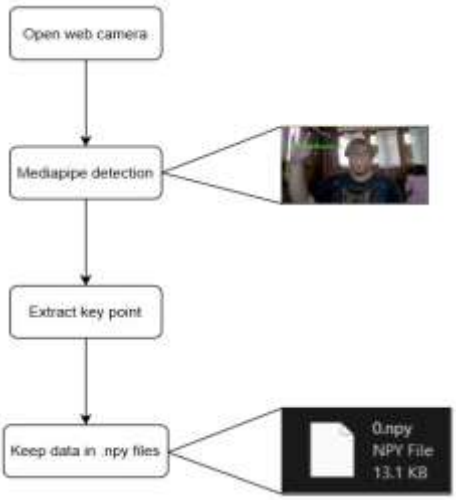
3.1 ขั้นตอนการจัดเตรียมข้อมูล

ภาพประกอบที่ 3.1 ตัวอย่างท่าทางภาษามือจาก www.th-sl.com

3.1.1 การเก็บข้อมูล

โครงงานนี้ได้รวบรวมข้อมูลและจัดทำวิดีโอเพื่อเก็บท่าทางภาษามือ โดยได้นำตัวอย่างท่าทางมาจากเว็บไซต์ th-sl.com ซึ่งเป็นเว็บไซต์ฐานข้อมูลภาษามือไทยที่น่าเชื่อถือ ซึ่งท่าทางที่นำมาใช้จะเป็นท่าทางที่ใช้ในชีวิตประจำวันทั้งหมด 20 ท่าทาง โดยแต่ละท่าทาง จะมี 160 วิดีโอ รวมทั้งหมดเป็น 3,200 วิดีโอ โดยการเก็บ Dataset จะเก็บไว้ในรูปแบบของไฟล์ .npy โดยก่อนจะเก็บข้อมูลเป็น .npy จะต้องทำการแยก Key point ที่ได้จากวิดีโอก่อน โดย Key point จะได้มาจากการแยกเฟรมของวิดีโอออกเป็น 30 เฟรม จากนั้นทำ Mediapipe detection เพื่อหาตำแหน่ง Landmark โดยจะมีส่วนของ มือซ้าย 21 จุด, มือขวา 21 จุด, ตัว 23 จุด (ตัดจุดที่ไม่จำเป็นออก 10 จุด จาก 33 จุด) และใบหน้า 468 จุด ในข้อมูลของ key point จะประกอบด้วยข้อมูลตำแหน่งแนวนอน x , ข้อมูลตำแหน่งแนวตั้ง y ,ข้อมูลความลึก z ,ข้อมูลการ

มองเห็น Visibility รวมแล้วทั้งหมดจะได้ Key point จำนวน 1,622 จุดApplication



ภาพประกอบที่ 3.2 ขั้นตอนการทำงานของการเก็บ Key point

จากภาพประกอบที่ 3.2 ขั้นตอนแรกคือการเปิดกล้องเว็บแคม ในที่นี้ เราจะใช้ cv2 เข้ามาช่วยในการเปิดกล้องเว็บแคม และเพื่อที่จะแยก Key point ออกมาได้นั้น จำเป็นต้องใช้ Mediapipe detection เพื่อหาตำแหน่ง Landmark ของแต่ละจุด

```
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 CONVERSION RGB 2 BGR
    return image, results
```

ภาพประกอบที่ 3.3 โค้ดการทำ Mediapipe Detection

จากภาพประกอบที่ 3.3 จะมีพารามิเตอร์คือเฟรมของวิดีโอและ Holistic model ซึ่งเป็นโมเดลที่ทำให้สามารถตรวจจับท่าทางมือ และใบหน้าของมนุษย์ได้ในเวลาเดียวกัน หลังจากส่งพารามิเตอร์เข้ามาแล้วจะทำการเปลี่ยนระบบสีจาก BGR เป็น RGB ก่อน จากนั้นจึงทำการทำนายเพื่อหาจุด Landmark แล้วจึงเปลี่ยนระบบสีจาก RGB กลับไปเป็น BGR เพื่อให้สามารถใช้งานกับ cv2 ได้

หลังจากที่ได้จุด Landmark ของเฟรมมาแล้ว ต่อไปก็จะเป็นการแยก Key points จากเฟรม แต่ในการที่จะนำข้อมูล Key point ไปฝึกฝนโมเดลได้นั้นจำเป็นต้องทำข้อมูลให้เป็น Vector ก่อน ดังภาพประกอบที่ 3.4 ดังนี้

ภาพประกอบที่ 3.4 โค้ดการทำ Extracting Key Points

จากภาพประกอบที่ 3.4 จะเป็นการนำค่าตำแหน่ง x, y, z และค่าการมองเห็น Visibility จากจุด Landmark ของ ตัว หน้า มือซ้าย และมือขวา มาเก็บเป็น numpy array แต่หากจุดไหนที่ตรวจจับไม่พบ จะกำหนดให้จุดนั้นเป็น 0 ทั้งหมด และหลังจากนั้น จึงจะนำเอาค่า Key points มา Concatenate หรือนำมารวมกันเพื่อให้ได้ซึ่งข้อมูลที่เป็น Vector มา

หลังจากได้ข้อมูล Key points มาแล้ว ก็จะเก็บข้อมูลลงไว้ที่ไฟล์ .npy เพื่อจะนำไปใช้ในการฝึกฝนโมเดลต่อไป โดยการเก็บข้อมูลลงไฟล์ .npy ทำได้ดังภาพประกอบที่ 3.5 ดังนี้

```
npy_path = os.path.join(DATA_PATH, action, str(sequence), str(frame_num))
np.save(npy_path, keypoints)
```

ภาพประกอบที่ 3.5 โค้ดการเก็บข้อมูลลงไฟล์ .npy

จากภาพประกอบที่ 3.5 จะเป็นการเก็บข้อมูลลงไฟล์ .npy โดยใช้ Library OS มาช่วยในการเก็บข้อมูลลงเครื่อง โดย

- DATA_PATH คือ ตำแหน่งของ folder ที่ต้องการจะเก็บ
- action คือ ตำแหน่งของชื่อท่าทางภาษามือ
- sequence คือ ตำแหน่งหมายเลขของวิดีโอ ซึ่งมีอยู่ทั้งหมด 150 วิดีโอต่อท่าทาง
- frame_num คือ ตำแหน่งของหมายเลขเฟรมที่แยกมาจากวิดีโอ ซึ่งกำหนดไว้เป็น 30 เฟรม

3.1.2 การทำ Data Preprocessing

การจะนำข้อมูลไปฝึกฝนได้นั้น ต้องทำการแยกข้อมูลออกเป็น 2 ส่วน ได้แก่ ข้อมูล Train และข้อมูล Validation โดยในที่นี้ จะแบ่งข้อมูล Train เป็น 80% และข้อมูล Validation เป็น 20% ซึ่งจะทำโดยการใช้คำสั่ง train_test_split ซึ่งเป็นคำสั่งในการแยกชุดข้อมูลที่ถูกสร้างขึ้นโดย sklearn

```
# Split data to x_train, y_train, x_test, y_test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

ภาพประกอบที่ 3.6 คำสั่งในการแยกชุดข้อมูล

```
x_train: (2560, 30, 1622)
y_train: (2560, 20)
x_test: (640, 30, 1622)
y_test: (640, 20)
```

ภาพประกอบที่ 3.7 ขนาดของชุดข้อมูลหลังการแยกชุดข้อมูล

โดยสัดส่วนของข้อมูลของแต่ละท่าหลังทำการแยกข้อมูล จะสามารถดูได้ตามตารางที่ 3.1 ดังนี้

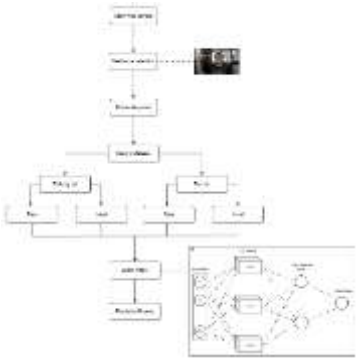
ตารางที่ 3.1 สัดส่วนของข้อมูลแต่ละท่า

ท่าทางภาษามือ	สัดส่วนของข้อมูล	
	Train	Validation
โกรธ	129	31
ไม่ชอบ	128	32
ง่าย	126	34
กิน	123	37
สบายดี	129	31
ลืม	130	30
โชคดี	128	32
ยาก	127	33

ตารางที่ 3.1 สัดส่วนของข้อมูลแต่ละท่า (ต่อ)

ท่าทางภาษามือ	สัดส่วนของข้อมูล	
	Train	Validation
ยาก	127	33
สวัสดี	125	35
ช่วยด้วย	135	25
หิว	131	29
ชอบ	127	33
รัก	129	31
ผู้ชาย	129	31
เศร้า	121	39
ป่วย	129	31
ขอโทษ	121	39
ขอบคุณ	127	28
เข้าใจ	134	33
คุณ	125	25

3.2 ขั้นตอนการพัฒนาโมเดล



ภาพประกอบที่ 3.8 ขั้นตอนการพัฒนาโมเดล

3.2.1 การสร้าง Model

หลังจากเตรียมข้อมูลและแยกชุดข้อมูลออกเป็นข้อมูล Train และ Validation แล้ว ก็พร้อมที่จะนำข้อมูลเหล่านี้ไปทำการฝึกฝนเพื่อให้ได้ซึ่งโมเดลการแปลภาษามือออกมา โดยการสร้างโมเดล จะสร้างได้ดังนี้

```
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,1622)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```

ภาพประกอบที่ 3.9 โค้ดในการสร้างโมเดลแต่ละ Layer

จากภาพประกอบที่ 3.8 โมเดลที่จะสร้างจะเริ่มจากการสร้าง Sequential เพื่อใช้เป็นตัวสร้าง Layer ที่ Stack กันเป็นเส้นตรงโดย

Layer แรกสำหรับการนำข้อมูลเข้าคือ LSTM layer ซึ่งมีจำนวน 64 ยูนิต และให้ Return ข้อมูลที่เป็นลำดับออกมา (โดยข้อมูลที่ได้ออกมานี้จะทำการส่งต่อไปยัง Layer ถัดไป) และ Activation function ที่ใช้ใน Layer นี้คือ ReLU ซึ่งใช้เพื่อคำนวณและประเมินประสิทธิภาพของผลลัพธ์ และขนาดของข้อมูลที่จะนำเข้าใน Layer นี้ต้องมีขนาด (30, 1622) ซึ่ง 30 คือจำนวนเฟรมของแต่ละวิดีโอ และ 1,622 คือจำนวนของตำแหน่ง Key points ในแต่ละเฟรม

Layer ที่สองเป็นอีก LSTM layer ซึ่งมีจำนวน 128 ยูนิต และให้ Return ข้อมูลที่เป็นลำดับออกมา และใช้ Activation function คือ ReLU

Layer ที่สามก็เป็นอีก LSTM layer ซึ่งมีจำนวน 64 ยูนิต แต่ใน Layer นี้จะไม่ให้ Return ข้อมูลที่เป็นลำดับออกมา และ Activation function ที่ใช้ใน Layer นี้คือ ReLU

Layer ที่สี่ คือ Dense หรือ Fully connected layer ซึ่งมีจำนวน 64 ยูนิต ใช้ ReLU เป็น Activation function และใช้ข้อมูลที่ส่งออกมาจาก Layer ก่อนหน้าเป็นข้อมูลที่ใช้นำเข้า

Layer ที่ห้า คือ Dense หรือ Fully connected layer ซึ่งมีจำนวน 32 ยูนิต และใช้ ReLU เป็น Activation function

Layer ที่หก คือ Dense หรือ Fully connected layer ซึ่งเป็น Layer สุดท้ายของโมเดลนี้ โดยเป็น Layer สุดท้ายที่จะส่งข้อมูลออกไป จำนวนยูนิตของ Layer นี้จะมีจำนวน 20 ยูนิต ซึ่งตรงกับจำนวนท่าทางที่โครงงานนี้ได้นำมาใช้ และจะใช้ Activation function คือ Softmax function ซึ่งจะเป็นฟังก์ชั่นที่ใช้สำหรับการจำแนกข้อมูลหลายคลาส โดยข้อมูลที่ส่งออกมาของ Layer นี้จะแสดงถึงความน่าจะเป็นของแต่ละคลาส

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 30, 64)            431872
_________________________________________________________________
lstm_1 (LSTM)                (None, 30, 128)           98816
_________________________________________________________________
lstm_2 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 20)                660
=================================================================
Total params: 586,996
Trainable params: 586,996
Non-trainable params: 0
_________________________________________________________________
```

ภาพประกอบที่ 3.10 จำนวนพารามิเตอร์ของโมเดลในแต่ละ Layer

```
model.compile(optimizer='Adam',
              loss='categorical_crossentropy',
              metrics=['categorical_accuracy'])
```

ภาพประกอบที่ 3.11 พารามิเตอร์ของ model.compile

จากภาพประกอบที่ 3.10 สำหรับในโมเดลนี้จะใช้ Optimizer คือ Adam เพื่อทำให้ค่า Loss ที่ออกมาในการฝึกฝนโมเดลมีค่าน้อยที่สุด และเลือกใช้ Categorical crossentropy มาใช้เป็น Loss function เพื่อวัดว่าโมเดลทำงานได้ดีเพียงใดในการฝึกฝน และยังใช้เพื่อปรับพารามิเตอร์ของโมเดลให้เหมาะสมระหว่าง

ทำการฝึกฝน โดย Categorical crossentropy จะนำมาใช้กับงานที่เป็น Multi-Class Classification ดังงานนี้ และสุดท้ายจะเลือกใช้ Metric คือ Categorical accuracy เพื่อใช้ในการประเมินประสิทธิภาพของโมเดลในขณะทำการฝึกฝนและทดสอบ โดย Categorical accuracy จะเป็นตัวที่จะคำนวณว่าเปอร์เซ็นต์ของค่าที่ทำนายนั้น ว่าตรงกับค่าที่เป็นจริงเป็นเท่าไหร่ โดยการนำจำนวนของข้อมูลที่ทายถูกไปหารกับจำนวนข้อมูลทั้งหมด

```
filepath = 'C:/Users/ASUS/Python/Project/Model_2/'          +
checkpoint = ModelCheckpoint(filepath=filepath,
                             monitor='val_categorical_accuracy',
                             verbose=1,
                             save_best_only=True,
                             mode='max',
                             save_weights_only=True)
```

ภาพประกอบที่ 3.12 Callbacks function

จากภาพประกอบที่ 3.12 Callback function ในโมเดลนี้จะใช้ ModelCheckpoint เพื่อกำหนดรอบ Epochs ที่ดีที่สุดในการฝึกฝนโมเดลแม้ว่าจะมีการเกิด Overtraining ขึ้นแล้วก็ตาม

3.3 ขั้นตอนการเรียนรู้และทดสอบโมเดล

หลังจากได้สร้างโมเดล LSTM เรียบร้อยแล้ว ต่อไปจะเป็นการกำหนดค่าเบื้องต้นของโมเดลเพื่อใช้ในการฝึกฝน โดยการฝึกฝนโมเดล จะอธิบายได้ดังต่อไปนี้

3.3.1 การฝึกฝนโมเดล

```
history = model.fit(X_train, y_train, epochs=200,
                    validation_data=(X_test, y_test),
                    callbacks=[checkpoint],
                    batch_size=64)
```

ภาพประกอบที่ 3.13 โค้ดการฝึกฝนโมเดล

จากภาพประกอบที่ 3.11 สามารถอธิบายพารามิเตอร์ของ model.fit ได้ดังนี้

- X_train คือ ข้อมูลตำแหน่งของ Key points ที่ใช้ในการฝึกฝน ซึ่งมีขนาดคือ (2560, 30, 1622)

- y_train คือ ข้อมูลชื่อท่าทางภาษามือที่ใช้ในการฝึกฝน ซึ่งมีขนาดคือ (2560, 20)

- epochs คือ ค่าที่ใช้กำหนดว่าจะทำการฝึกฝนโมเดลกี่รอบ ในที่นี้กำหนดไว้ 200 รอบ

- validation_data คือ พารามิเตอรที่เก็บค่า x และ y ของข้อมูลทดสอบ

- X_test คือ ข้อมูลตำแหน่งของ Key points ที่ใช้ในการทดสอบ ซึ่งมีขนาดคือ (640, 30, 1622)

- y_test คือ ข้อมูลชื่อท่าทางภาษามือที่ใช้ในการทดสอบ ซึ่งมีขนาดคือ (640, 20)

- callbacks คือ ฟังก์ชันที่ใช้เพื่อกำหนดรอบที่ดีที่สุดในการฝึกฝนข้อมูล ซึ่งได้เลือกใช้ ModelCheckpoint

- batch_size คือ จำนวนรายการข้อมูลที่จะให้ Optimizer คำนวณในหนึ่งครั้ง ซึ่งกำหนดเป็น 64

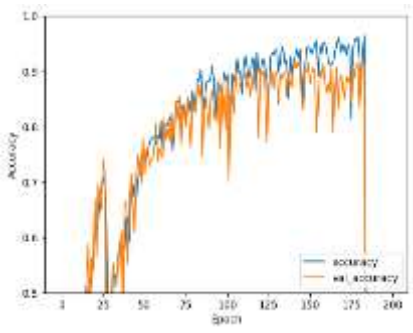
หลังจากกำหนดค่าเบื้องต้นเสร็จแล้ว ก็จะทำการฝึกฝนโมเดล เพื่อให้ได้มาซึ่งโมเดลการแปลท่าทางภาษามือ

```
Epoch 00141: val_categorical_accuracy did not improve from 0.90469
Epoch 142/200
40/40 [==============================] - 3s 80ms/step - loss: 0.2253 - categori
cal_accuracy: 0.9234 - val_loss: 0.3692 - val_categorical_accuracy: 0.9203

Epoch 00142: val_categorical_accuracy improved from 0.90469 to 0.92031, saving
model to C:/Users/ASUS/Python/Project/Model_2\
```

ภาพประกอบที่ 3.14 การฝึกฝนข้อมูลรอบที่ดีที่สุด

จากภาพประกอบที่ 3.14 ในการฝึกฝนข้อมูลในแต่ละรอบ จะแสดงค่า Loss, Categorical accuracy, Validation loss และ Validation accuracy ของในแต่ละรอบ โดยการดูประสิทธิภาพของโมเดลนั้นจะดูได้จากค่า Validation loss ซึ่งหากยิ่งน้อยจะหมายความว่าโมเดลนั้นมีข้อมูลที่สูญหายจากจากการฝึกฝนน้อย และค่า Validation accuracy ซึ่งหากยิ่งมีค่าสูงจะหมายความว่าโมเดลมีประสิทธิภาพที่ดี ซึ่งในการฝึกฝนโมเดลในรอบที่ดีที่สุดหรือรอบที่ 142 ค่า Validation loss และ Validation accuracy อยู่ที่ 0.3692 และ 0.92 ตามลำดับ

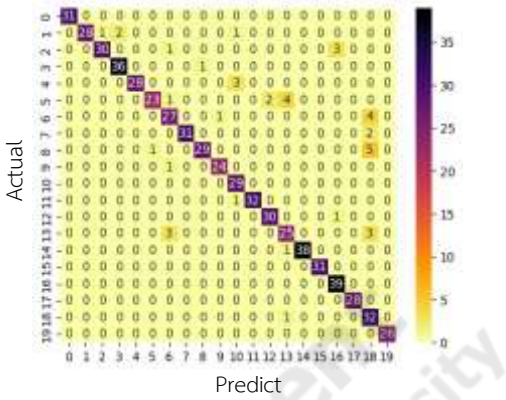


ภาพประกอบที่ 3.15 กราฟผลการฝึกฝนโมเดล

3.4 การวัดประสิทธิภาพของโมเดล

ในขั้นตอนการวัดประสิทธิภาพจะใช้โมเดลจำแนกท่าทางภาษามือตามชุดข้อมูลที่ได้จัดเตรียมไว้ คือ 3000 วิดีโอ แบ่งเป็นท่าทางละ 150 วิดีโอและทำการแยกข้อมูล 2 ส่วนคือข้อมูล Train และ Validation คำตอบที่เป็นแนวทแยงนั้นคือผลลัพธ์ของจำนวนที่โมเดลทำนายถูกต้อง โดยจะมีท่าทางตามลำดับดังนี้

- 0 : โกรธ
- 1 : ไม่ชอบ
- 2 : ง่าย
- 3 : กิน
- 4 : สบายดี
- 5 : ลืม
- 6 : โชคดี
- 7 : ยาก
- 8 : สวัสดี
- 9 : ช่วยด้วย
- 10 : หิว
- 11 : ชอบ
- 12 : รัก
- 13 : ผู้ชาย
- 14 : เสียใจ
- 15 : ป่วย
- 16 : ขอโทษ
- 17 : ขอบคุณ
- 18 : เข้าใจ
- 19 : คุณ



ภาพประกอบที่ 3.16 Confusion Matrix

4. ผลการทดลอง

ในบทนี้จะเป็นการแสดงขั้นตอนในการทดสอบกระบวนการการใช้งานโมเดลการแปลภาษามือด้วยการเรียนรู้เชิงลึก (Sign Language Translation Using Deep Learning) ซึ่งจะแสดงถึงผลการทดลองระหว่างโมเดลตัวเดิมที่เก็บข้อมูลการฝึกฝนและข้อมูลการทดสอบมาจากผู้ใช้เพียง 2 คน และโมเดลตัวใหม่ที่เก็บข้อมูลการฝึกฝนและข้อมูลการทดสอบมาจากผู้ใช้ 7 คน และทำการลดจำนวนของ Key points ที่เกินมา โดยจะประเมินประสิทธิภาพของโมเดลทั้งสองด้วย Confusion matrix

4.1 การเปรียบเทียบประสิทธิภาพของโมเดล

ในการแปลภาษามือด้วยการเรียนรู้เชิงลึก ผู้ใช้ได้เลือกใช้สถาปัตยกรรม LSTM หรือ Long-Short Term Memory มาใช้ในการพัฒนาโมเดล โดยได้พัฒนาโมเดลขึ้นมา 2 ตัว โดยในโมเดลตัวแรก ได้ทำการเก็บข้อมูล Key points ทั้งหมด 1,662 จุด จากผู้ใช้ 2 คน และโมเดลตัวที่ 2 ที่ได้การพัฒนาให้ดีขึ้นยิ่งขึ้น ด้วยการตัด Key points ในส่วนที่ไม่จำเป็นออก จนเหลือเพียง 1,622 จุด และเก็บข้อมูลจากผู้ใช้ทั้งหมด 7 คน

การประเมินประสิทธิภาพของแต่ละ

ท่าทางภาษามือของทั้งสองโมเดล จะทำการประเมินจากการหา ค่าความแม่นยำ (Precision) ค่าความระลึก (Recall) และค่าความถูกต้อง (Accuracy) ด้วย Confusion matrix

4.1.1 โมเดลตัวที่หนึ่ง

ทำการเก็บข้อมูลมาจากผู้ใช้ 2 คน โดยเก็บมาท่าทางละ 150 วิดีโอ รวมทั้งสิ้น 3,000 วิดีโอ แล้วทำการแบ่งข้อมูลทดสอบออกมาจากข้อมูลทั้งหมด 20% โดยในแต่ละวิดีโอได้ทำการแยกออกเป็น 30 เฟรม แล้วเก็บตำแหน่ง Key points มาเฟรมละ 1,662 จุด

โดยในการทำการประเมินประสิทธิภาพของโมเดล ได้ใช้ข้อมูลทดสอบที่แบ่งออกมาจากข้อมูลทั้งหมด 20% หรือคิดเป็น 600 วิดีโอจากทั้งหมด มาเพื่อใช้ในการวัดประสิทธิภาพของโมเดล

ตารางที่ 4.1 ผลการประเมินประสิทธิภาพของโมเดลตัวแรก

ท่าทางภาษามือ	Precision	Recall	F1-Score
โกรธ	0.91	1.0	0.95
ไม่ชอบ	1.0	1.0	1.0
ง่าย	0.84	1.0	0.91
กิน	1.0	0.96	0.98
สบายดี	0.92	0.88	0.90
ลืม	1.0	1.0	1.0
โชคดี	0.95	0.85	0.90
ยาก	0.91	1.0	0.95
สวัสดี	1.0	1.0	1.0
ช่วยด้วย	0.93	0.90	0.92
หิว	0.97	0.97	0.97
ชอบ	0.96	0.96	0.96
รัก	1.0	0.97	0.98

ตารางที่ 4.1 ผลการประเมินประสิทธิภาพของโมเดลตัวแรก (ต่อ)

ท่าทางภาษามือ	Precision	Recall	F1-Score
ผู้ชาย	0.89	0.91	0.90
เศร้า	1.0	0.88	0.93
ป่วย	1.0	1.0	1.0
ขอโทษ	0.95	0.78	0.86
ขอบคุณ	1.0	0.92	0.96
เข้าใจ	0.82	0.95	0.88
คุณ	1.0	1.0	1.0
Accuracy	0.95		

คลาสที่มีค่า Precision น้อยที่สุด คือ ท่าเข้าใจ ซึ่งอยู่ที่ 0.82 สาเหตุอาจเกิดมาจากตำแหน่งของ Key points ของท่านี้ ที่ไปคล้ายคลึงกับ ท่าช่วยด้วยและท่าผู้ชาย เลยอาจทำให้โมเดลทำนายผิดกลายเป็นท่าทางที่ได้กล่าวมา

คลาสที่มีค่า Precision มากที่สุด คือ ท่าไม่ชอบ ท่ากิน ท่าลืม ท่าสวัสดี ท่ารัก ท่าเศร้า ท่าป่วย ท่าขอบคุณ และ ท่าคุณ ซึ่งอยู่ที่ 1.0 สาเหตุเกิดจากในการทำนายผลของโมเดล โมเดลได้ทำนายว่าท่าทางเหล่านี้นั้นถูกทั้งหมด เช่น มีชุดข้อมูลจริงของท่ากินอยู่ 31 วิดีโอ และโมเดลได้ทำนายว่าเป็นท่ากินทั้งหมด 30 วิดีโอ ซึ่งถูกทั้งหมด แต่โมเดลไปทายว่าอีก 1 วิดีโอที่เหลือของท่ากินเป็นท่าทางอื่น หรือในการเก็บชุดข้อมูล มีการเก็บจากผู้ใช้เพียง 2 คนเท่านั้น จึงทำให้ในชุดข้อมูล ไม่มีความแตกต่างของข้อมูลมากเท่าที่ควร

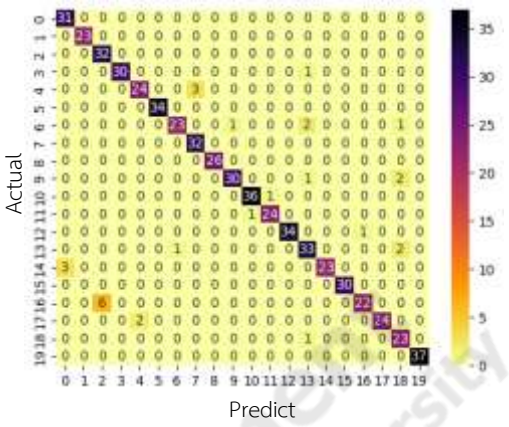
คลาสที่มีค่า Recall น้อยที่สุด คือ ท่าขอโทษ ซึ่งอยู่ที่ 0.78 สาเหตุอาจเกิดมาจากตำแหน่ง Key points ของท่านี้มีลักษณะที่ใกล้เคียงกับท่าง่าย จึงทำให้ตัวโมเดลทำนายผิด

ว่าเป็นท่าง่ายในหลายวิดีโอ

คลาสที่มีค่า Recall มากที่สุด คือ ท่าโกรธ ท่าไม่ชอบ ท่าง่าย ท่าลืม ท่ายาก ท่าสวัสดี ท่าป่วย และ ท่าคุณ ซึ่งอยู่ที่ 1.0 สาเหตุเกิดจากในการทำนายผลของโมเดล โมเดลได้ทำนายว่าท่าทางเหล่านี้นั้นถูกทั้งหมดตามจำนวนของข้อมูลที่เป็นจริง เช่น ท่าโกรธ มีชุดข้อมูลจริงอยู่ 31 วิดีโอ และโมเดลได้ทำนายว่าเป็นท่าโกรธทั้งหมด 34 วิดีโอ ซึ่งถูกครบทั้ง 31 วิดีโอตามข้อมูลจริง แต่มีอีก 3 วิดีโอที่โมเดลได้ทำนายว่าเป็นท่าโกรธ แต่ไม่ถูกต้อง เลยทำให้ค่า Recall มีค่าเป็น 1.0 หรือในอีกกรณี ในการเก็บชุดข้อมูล มีการเก็บจากผู้ใช้เพียง 2 คนเท่านั้น จึงทำให้ในชุดข้อมูล ไม่มีความแตกต่างของข้อมูลมากเท่าที่ควร

คลาสที่มีค่า F1-Score น้อยที่สุด คือ ท่าขอโทษ ซึ่งอยู่ที่ 0.86 เนื่องจากท่าขอโทษ มีค่า Recall ที่ค่อนข้างต่ำ ซึ่งอาจเป็นผลมาจากตำแหน่ง Key points ของท่านี้มีลักษณะที่ใกล้เคียงกับท่าง่าย จึงทำให้ตัวโมเดลทำนายผิดว่าเป็นท่าง่ายในหลายวิดีโอ

คลาสที่มีค่า F1-Score มากที่สุด คือ ท่าไม่ชอบ ท่าลืม ท่าสวัสดี ท่าป่วย และ ท่าคุณ ซึ่งอยู่ที่ 1.0 เพราะท่าทางเหล่านี้เป็นท่าทางที่มีตำแหน่ง Key points ค่อนข้างที่จะแตกต่างจากท่าทางอื่นๆ และนอกจากนี้ ยังมีค่า Precision และ Recall ที่สูงมากๆ แต่ในอีกกรณี ในการเก็บชุดข้อมูล มีการเก็บจากผู้ใช้เพียง 2 คนเท่านั้น จึงทำให้ในชุดข้อมูล ไม่มีความแตกต่างของข้อมูลมากเท่าที่ควร



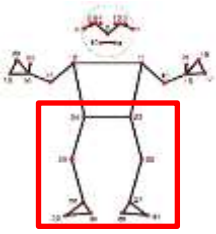
ภาพประกอบที่ 4.1 Confusion Matrix ของโมเดลตัวแรก

จาก Confusion Matrix ของโมเดลตัวแรก พบว่ามี Accuracy อยู่ที่ 0.95 แต่จากการใช้งานจริง พบว่า โมเดลไม่แม่นเท่าที่ควร สาเหตุที่การวัดประสิทธิภาพของโมเดลออกมาดีคาดว่าเกิดจากการใช้ข้อมูลทดสอบที่เก็บมาจากผู้ใช้เพียง 2 คนเท่านั้น จึงทำให้ข้อมูลไม่มีความหลากหลายเท่าที่ควร จึงทำให้เกิดการ Overfitting เกิดขึ้น

4.1.2 โมเดลตัวที่สอง

ทำการเก็บข้อมูลมาจากผู้ใช้ 7 คน โดยเก็บมาท่าทางละ 160 วิดีโอ รวมทั้งสิ้น 3,200 วิดีโอ แล้วทำการแบ่งข้อมูลทดสอบออกมาจากข้อมูลทั้งหมด 20% โดยในแต่ละวิดีโอได้ทำการแยกออกเป็น 30 เฟรม แล้วเก็บตำแหน่ง Key points มาเฟรมละ 1,622 จุด โดยได้ทำการตัด Key points ในส่วนของ Pose บางส่วนที่ไม่จำเป็นออก

โดยในการทำการประเมินประสิทธิภาพของโมเดล ได้ใช้ข้อมูลทดสอบที่แบ่งออกมาจากข้อมูลทั้งหมด 20% หรือคิดเป็น 640 วิดีโอจากทั้งหมด มาเพื่อใช้ในการวัดประสิทธิภาพของโมเดล



ภาพประกอบที่ 4.2 ส่วนที่ถูกตัดออก

ตารางที่ 4.2 ผลการประเมินประสิทธิภาพของเดลตัวที่สอง

ท่าทางภาษามือ	Precision	Recall	F1-Score
โกรธ	1.0	1.0	1.0
ไม่ชอบ	1.0	0.87	0.93
ง่าย	0.96	0.88	0.92
กิน	0.94	0.97	0.95
สบายดี	1.0	0.90	0.94
ลืม	0.95	0.76	0.85
โชคดี	0.81	0.84	0.83
ยาก	1.0	0.93	0.96
สวัสดี	0.96	0.82	0.89
ช่วยด้วย	0.96	0.96	0.96
หิว	0.85	1.0	0.92
ชอบ	1.0	0.96	0.98
รัก	0.93	0.96	0.95
ผู้ชาย	0.80	0.80	0.80
เศร้า	1.0	0.97	0.98
ป่วย	1.0	1.0	1.0
ขอโทษ	0.90	1.0	0.95
ขอบคุณ	1.0	0.90	0.95
เข้าใจ	0.69	0.96	0.81
คุณ	1.0	1.0	1.0
Accuracy	0.93		

คลาสที่มีค่า Precision น้อยที่สุด คือ ท่าเข้าใจ ซึ่งอยู่ที่ 0.69 สาเหตุเพราะ ตำแหน่ง Key points ของ ท่าเข้าใจ มีตำแหน่งที่ใกล้เคียงกับท่าทางอื่นๆ เช่น ท่าโชคดี ท่าผู้ชาย เป็นต้น จึงทำให้โมเดลทำนายผลผิดได้ง่าย

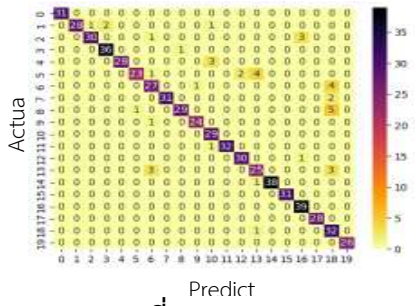
คลาสที่มีค่า Precision มากที่สุด คือ ท่าโกรธ ท่าไม่ชอบ ท่าสบายดี ท่ายาก ท่าชอบ ท่าเศร้า ท่าป่วย ท่าขอบคุณ และ ท่าคุณ ซึ่งอยู่ที่ 1.0 เพราะ ท่าทางเหล่านี้เป็นท่าที่มีตำแหน่งของ Key points ชัดเจน และค่อนข้างแตกต่างจากท่าทางอื่นๆ จึงทำให้โมเดลสามารถเรียนรู้ท่าทางเหล่านี้ได้ดี

คลาสที่มีค่า Recall น้อยที่สุดคือ ท่าลืม ซึ่งอยู่ที่ 0.76 สาเหตุเกิดมาจากในการทำนายโมเดล ท่าลืมเป็นท่าที่มี False Negative มากที่สุด หรือก็คือ โมเดลทำนายข้อมูลที่นำเข้ามาทดสอบผิดว่าเป็นท่าทางอื่น ซึ่งสาเหตุหลักๆเกิดมาจากข้อมูลทดสอบที่นำมาใช้ได้เก็บมาจากผู้ใช้งานหลายคน และอาจมีบางคนที่ทำท่าทางได้ไม่ถูกต้องเท่าที่ควร จึงทำให้โมเดลทำนายผลผิด

คลาสที่มีค่า Recall มากที่สุด คือ ท่าโกรธ ท่าหิว ท่าป่วย ท่าขอโทษ ท่าขอบคุณ และท่าคุณ ซึ่งอยู่ที่ 1.0 เพราะ ท่าทางเหล่านี้ล้วนแต่เป็นท่าทางที่มีตำแหน่ง Key points ค่อนข้างที่จะแตกต่างจากท่าอื่นๆ จึงทำให้โมเดลสามารถจำแนกคุณลักษณะได้ง่าย จึงทำให้ท่าทางเหล่านี้นั้นค่อนข้างที่จะมีประสิทธิภาพที่ดี

คลาสที่มีค่า F1-Score น้อยที่สุด คือ ท่าผู้ชาย ซึ่งอยู่ที่ 0.80 เพราะ ท่าผู้ชาย มีค่า Precision และ Recall ที่ค่อนข้างน้อย เนื่องมาจากตำแหน่ง Key points ของท่าผู้ชายนั้นค่อนข้างใกล้เคียงกับ ท่าโชคดี และ ท่าลืม จึงทำให้โมเดลทำนายผลออกมาผิดบางส่วน

คลาสที่มีค่า F1-Score มากที่สุด คือ ท่าโกรธ ท่าป่วย ท่าขอบคุณ และ ท่าคุณ เพราะ ท่าทางเหล่านี้ล้วนเป็นท่าทางที่ตำแหน่ง Key points ค่อนข้างแตกต่างจากท่าทางอื่นๆ จึงทำให้โมเดลสามารถทำนายผลออกมาได้ถูกต้องและแม่นยำ



ภาพประกอบที่ 4.3 Confusion Matrix ของโมเดลตัวที่สอง

จากภาพประกอบที่ 4.3 พบว่ามี Accuracy อยู่ที่ 0.93 ซึ่งน้อยกว่าโมเดลตัวแรกที่มีค่า Accuracy อยู่ที่ 0.95 แต่จากการใช้งานจริง พบว่า โมเดลตัวที่สองมีการทำนายผลที่แม่นยำกว่าโมเดลตัวแรก ของโมเดลตัวแรกนั้นเกิดการ Overfitting เนื่องจากข้อมูลที่นำมาใช้ในการฝึกฝนโมเดลไม่มีความหลากหลายเท่าที่ควร ดังนั้น จึงได้เลือกโมเดลตัวที่สองมาใช้ในการแปลท่าทางภาษามือ

ตารางที่ 4.3 ตัวอย่างผลการทำนาย

ผลการทำนาย		ผลเฉลย
โมเดลตัวแรก	**โมเดลตัวที่สอง**	
โกรธ	โกรธ	โกรธ
ไม่ชอบ	ไม่ชอบ	ไม่ชอบ
ขอโทษ	ง่าย	ง่าย
หิว	กิน	กิน
สบายดี	สบายดี	สบายดี
ลืม	ลืม	ลืม
ผู้ชาย	โชคดี	โชคดี
ชอบ	ชอบ	ชอบ

ตารางที่ 4.3 ตัวอย่างผลการทำนาย (ต่อ)

ผลการทำนาย		ผลเฉลย
โมเดลตัวแรก	**โมเดลตัวที่สอง**	
รัก	รัก	รัก
ผู้ชาย	ผู้ชาย	ผู้ชาย
เศร้า	เศร้า	เศร้า
ป่วย	ป่วย	ป่วย
ขอโทษ	ขอโทษ	ขอโทษ
สบายดี	ขอบคุณ	ขอบคุณ
โชคดี	เข้าใจ	เข้าใจ
คุณ	คุณ	คุณ
ขอโทษ	ยาก	ยาก
สวัสดี	สวัสดี	สวัสดี
ช่วยด้วย	ช่วยด้วย	ช่วยด้วย
หิว	หิว	หิว

4.2 การวัดประสิทธิภาพของ Desktop Application

จากการทดลองใช้งานโมเดลบนตัวแอปพลิเคชัน ผู้จัดทำได้ทำการทดสอบกับผู้ใช้ 7 คน เพื่อทำการวัดประสิทธิภาพของตัวแอปพลิเคชัน โดยผลการทดสอบ มีดังนี้

4.2.1 ผลการทดสอบแอปพลิเคชันจากผู้ใช้ทั้ง 7 คน

4.2.1.1 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 1

โดยผลการทดสอบ มีดังนี้

ตารางที่ 4.4 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 1

ท่าทางที่	ผลทำนาย	ผลเฉลย
1	โกรธ	โกรธ
2	ไม่ชอบ	ไม่ชอบ
3	ขอโทษ	ง่าย
4	กิน	กิน
5	สบายดี	สบายดี
6	ลืม	ลืม
7	โชคดี	โชคดี
8	ยาก	ยาก
9	สวัสดี	สวัสดี
10	คุณ	ช่วยด้วย
11	หิว	หิว
12	ชอบ	ชอบ
13	รัก	รัก
14	ผู้ชาย	ผู้ชาย
15	เศร้า	เศร้า
16	ลืม	ป่วย
17	ขอโทษ	ขอโทษ
18	ขอบคุณ	ขอบคุณ
19	ผู้ชาย	เข้าใจ
20	คุณ	คุณ

4.2.1.2 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 2

ตารางที่ 4.5 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 2

ท่าทางที่	ผลทำนาย	ผลเฉลย
1	โกรธ	โกรธ
2	ไม่ชอบ	ไม่ชอบ

ตารางที่ 4.5 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 2 (ต่อ)

ท่าทางที่	ผลทำนาย	ผลเฉลย
3	ขอโทษ	ง่าย
4	กิน	กิน
5	สบายดี	สบายดี
6	ลืม	ลืม
7	ช่วยด้วย	โชคดี
8	รัก	ยาก
9	สวัสดี	สวัสดี
10	ช่วยด้วย	ช่วยด้วย
11	ไม่ชอบ	หิว
12	ชอบ	ชอบ
13	รัก	รัก
14	ผู้ชาย	ผู้ชาย
15	เศร้า	เศร้า
16	ลืม	ป่วย
17	ขอโทษ	ขอโทษ
18	ขอบตุณ	ขอบคุณ
19	เข้าใจ	เข้าใจ
20	คุณ	คุณ

4.2.1.3 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 3

ตารางที่ 4.6 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 3

ท่าทางที่	ผลทำนาย	ผลเฉลย
1	ผู้ชาย	โกรธ
2	ไม่ชอบ	ไม่ชอบ
3	ง่าย	ง่าย
4	กิน	กิน

ตารางที่ 4.6 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 3 (ต่อ)

ท่าทางที่	ผลทำนาย	ผลเฉลย
5	สบายดี	สบายดี
6	ลืม	ลืม
7	คุณ	โชคดี
8	ยาก	ยาก
9	รัก	สวัสดี
10	ช่วยด้วย	ช่วยด้วย
11	หิว	หิว
12	ชอบ	ชอบ
13	รัก	รัก
14	ผู้ชาย	ผู้ชาย
15	เศร้า	เศร้า
16	ลืม	ป่วย
17	ขอโทษ	ขอโทษ
18	ขอบคุณ	ขอบคุณ
19	เข้าใจ	เข้าใจ
20	คุณ	คุณ

4.2.1.4 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 4

ตารางที่ 4.7 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 4

ท่าทางที่	ผลทำนาย	ผลเฉลย
1	โกรธ	โกรธ
2	ไม่ชอบ	ไม่ชอบ
3	ขอโทษ	ง่าย
4	กิน	กิน
5	สบายดี	สบายดี
6	ลืม	ลืม

ตารางที่ 4.7 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 4 **(ต่อ)**

ท่าทางที่	ผลทำนาย	ผลเฉลย
7	เข้าใจ	โชคดี
8	ยาก	ยาก
9	สวัสดี	สวัสดี
10	ช่วยด้วย	ช่วยด้วย
11	หิว	หิว
12	ไม่ชอบ	ชอบ
13	รัก	รัก
14	ผู้ชาย	ผู้ชาย
15	เศร้า	เศร้า
16	ป่วย	ป่วย
17	ขอโทษ	ขอโทษ
18	ขอบคุณ	ขอบคุณ
19	เข้าใจ	เข้าใจ
20	คุณ	คุณ

4.2.1.5 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 5

ตารางที่ 4.8 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 5

ท่าทางที่	ผลทำนาย	ผลเฉลย
1	โกรธ	โกรธ
2	ไม่ชอบ	ไม่ชอบ
3	ง่าย	ง่าย
4	กิน	กิน
5	สบายดี	สบายดี
6	ลืม	ลืม
7	โชคดี	โชคดี
8	ยาก	ยาก

ตารางที่ 4.8 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 5 **(ต่อ)**

ท่าทางที่	ผลทำนาย	ผลเฉลย
9	รัก	สวัสดี
10	ช่วยด้วย	ช่วยด้วย
11	หิว	หิว
12	ชอบ	ชอบ
13	สวัสดี	รัก
14	ผู้ชาย	ผู้ชาย
15	เศร้า	เศร้า
16	ป่วย	ป่วย
17	ขอโทษ	ขอโทษ
18	ขอบคุณ	ขอบคุณ
19	เข้าใจ	เข้าใจ
20	คุณ	คุณ

4.2.1.6 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 6

ตารางที่ 4.9 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 6

ท่าทางที่	ผลทำนาย	ผลเฉลย
1	โกรธ	โกรธ
2	ไม่ชอบ	ไม่ชอบ
3	ขอโทษ	ง่าย
4	กิน	กิน
5	ขอบคุณ	สบายดี
6	ลืม	ลืม
7	ช่วยด้วย	โชคดี
8	ยาก	ยาก
9	สวัสดี	สวัสดี
10	ช่วยด้วย	ช่วยด้วย

ตารางที่ 4.9 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 6 (ต่อ)

ท่าทางที่	ผลทำนาย	ผลเฉลย
11	หิว	หิว
12	ไม่ชอบ	ชอบ
13	รัก	รัก
14	ผู้ชาย	ผู้ชาย
15	เศร้า	เศร้า
16	ลืม	ป่วย
17	ขอโทษ	ขอโทษ
18	สบายดี	ขอบคุณ
19	ผู้ชาย	เข้าใจ
20	คุณ	คุณ

4.2.1.7 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 7

ตารางที่ 4.10 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 7

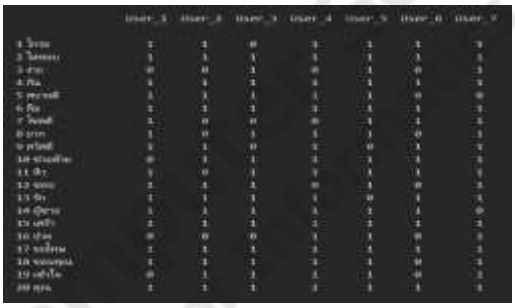
ท่าทางที่	ผลทำนาย	ผลเฉลย
1	โกรธ	โกรธ
2	ไม่ชอบ	ไม่ชอบ
3	ง่าย	ง่าย
4	กิน	กิน
5	ขอบคุณ	สบายดี
6	ลืม	ลืม
7	โชคดี	โชคดี
8	ยาก	ยาก
9	สวัสดี	สวัสดี
10	ช่วยด้วย	ช่วยด้วย
11	หิว	หิว
12	ชอบ	ชอบ

ตารางที่ 4.10 ผลการทดสอบแอปพลิเคชันของผู้ใช้คนที่ 7 (ต่อ)

ท่าทางที่	ผลทำนาย	ผลเฉลย
13	รัก	รัก
14	โชคดี	ผู้ชาย
15	เศร้า	เศร้า
16	ป่วย	ป่วย
17	ขอโทษ	ขอโทษ
18	ขอบคุณ	ขอบคุณ
19	เข้าใจ	เข้าใจ
20	คุณ	คุณ

4.2.2 สรุปและวิเคราะห์ผลการทดสอบแอปพลิเคชัน

จากผลการทดสอบแอปพลิเคชันของผู้ใช้ทั้งหมด 7 คน จะสรุปได้ดังภาพประกอบที่ 4.4 สรุปผลการทดลองใช้แอปพลิเคชันของผู้ใช้ทั้ง 7 คน ดังต่อไปนี้



ภาพประกอบที่ 4.4 สรุปผลการทดลองใช้แอปพลิเคชันของผู้ใช้ทั้ง 7 คน

โดยเลข 0 คือ ผลการทำนายที่ทำนายผิด และ เลข 1 คือผลการทำนายที่ทายถูกต้อง และจากในตารางข้างต้น มีผลที่ทำนายถูกต้องรวมกันทั้งหมดคือ 113 ครั้ง จากทั้งหมด 140 ครั้งดังนั้นจึงทำการคำนวณหาค่าเฉลี่ยของการทำนายผลที่ถูกต้อง ได้ดังนี้

$$Mean = \frac{113}{140} = 0.80$$

สรุปได้ว่า จากผลการทดลองการใช้งานจริงของแอปพลิเคชันโดยผู้ใช้ทั้ง 7 คน ตัวโมเดลที่ใช้บนแอปพลิเคชันมีค่าเฉลี่ยของการทำนายผลที่ถูกต้อง อยู่ที่ 0.80

และจากการวิเคราะห์ผลของแอปพลิเคชัน พบว่า มีท่าง่ายและท่าป่วย ที่มีการทำนายผิดเยอะที่สุด โดยทำนายผิดทั้งหมด 4 ครั้ง จาก 7 ครั้ง โดยทำนายผิดจากท่าง่าย เป็นท่าขอโทษ และจากท่าป่วย ทำนายผิดเป็นท่าลืม สาเหตุหลักๆ เป็นเพราะ ตำแหน่งของ Key points ของท่าง่ายนั้น มีตำแหน่งที่คล้ายกับท่าขอโทษ และ ตำแหน่งของท่าป่วยมีตำแหน่งที่ใกล้เคียงกับท่าลืม

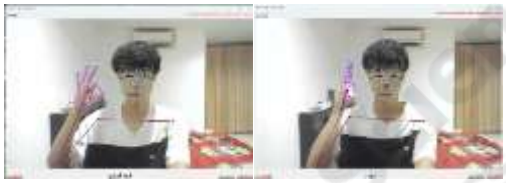
และยังพบว่า มีท่าไม่ชอบ ท่ากิน ท่าลืม ท่าเศร้า ท่าขอโทษ และท่าคุณ ที่มีผลการทำนายถูกต้องทั้งหมด ซึ่งสาเหตุที่มีการทำนายผลได้ถูกต้องทั้งหมด มีผลมาจากในท่าทางเหล่านี้นั้น มีตำแหน่ง Key points ที่ค่อนข้างแตกต่างจากท่าอื่นๆ จึงทำให้โมเดลมีการรู้จำที่ค่อนข้างดี

นอกจากนี้ยังมีผลการวิเคราะห์ในเรื่องของแสง เงา และ พื้นหลังของผู้ใช้ได้ดังนี้ ซึ่งมีผลกระทบโดยตรงต่อการทำนายของโมเดลบนแอปพลิชันอีกด้วย โดยในการใช้ Mediapipe detection ผู้จัดทำได้ทำการเรียกใช้ Holistic model ของทาง Mediapipe ซึ่งโมเดลที่ใช้ในการตรวจจับ Key points บริเวณ หน้า มือ และลำตัว ซึ่งมีข้อด้อยคือ ในการตรวจจับ หากตรวจจับในที่ที่แสงไม่เพียงพอหรือมีเงามาบดบังที่ผู้ใช้ และ พื้นหลังที่เป็นพื้นหลังที่ค่อนข้างจะมีลวดลายเยอะจนเกินไป จะทำให้ผลของการตรวจจับ Key points ออกมาผิดเพี้ยนได้ ซึ่งจะส่งผลให้ผลการทำนายของโมเดลการแปลท่าทางภาษามือนั้น ทำนายออกมาผิดพลาด

และการวิเคราะห์ในเรื่องของการทำนายผลในแอปพลิเคชัน พบว่า ในขณะใช้งานจริง มีการทำนายท่าทางทางออกมาหลาย

ท่าทาง แม้ผู้ใช้จะแสดงท่าทางเพียงท่าทางเดียว สาเหตุพบว่า เนื่องจากในการส่งข้อมูล Key points เข้าไปเพื่อให้โมเดลทำการทำนาย มีการส่งข้อมูลเข้าไปเรื่อย ๆ เนื่องจากการทำนายผลของแอปพลิเคชันเป็นการทำนายผลแบบ Real-time ดังนั้นจึงทำให้โมเดลมีการทำนายผลออกมาเรื่อย ๆ พร้อมกับการขยับท่าทางของผู้ใช้ ซึ่งอาจจะไปตรงกับท่าทางอื่นๆ ในโมเดล และในการทำนายผล ตัวโมเดลจะทำนายได้ถูกต้องและแม่นยำที่สุดเมื่อผู้ใช้ได้แสดงท่าทางที่ผู้ใช้ต้องการจะแสดงนั้น ค้างไว้ โดยไม่ขยับไปไหน เนื่องจากจะเป็นการส่งตำแหน่ง Key points ที่ค่อนข้างมั่นคงเข้าไปให้โมเดลทำการทำนายผล

4.3 ตัวอย่างการทำนายผลบน Desktop Application



ภาพประกอบที่ 4.5 ตัวอย่างการทำนายผลบน Desktop Application

4.4 สรุปและวิเคราะห์ผลการทดลอง

จากการทดลองในการสร้างโมเดลการแปลภาษามือด้วยการเรียนรู้เชิงลึก โดยใช้สถาปัตยกรรม LSTM โดยไดทำการทดลองจากชุดข้อมูลทั้งหมด 3,000 วิดีโอในโมเดลตัวแรก และ 3,200 วิดีโอในโมเดลตัวที่สอง ได้ผลสรุปว่า โมเดลตัวแรกมีค่า Accuraccy อยู่ที่ 95 เปอร์เซ็น แต่เกิดการ Overfitting เกิดขึ้น เนื่องจากในการเก็บข้อมูลได้เก็บมาจากผู้ใช้เพียง 2 คน ทำให้ข้อมูลไม่ได้มีความหลากหลายเท่าที่ควร และโมเดลตัวที่สองมีค่า Accuracy อยู่ที่ 93 เปอร์เซ็น และในการใช้งานจริง โมเดลตัวที่สองมีประสิทธิภาพกับข้อมูลที่ไม่เคยเจอได้ดีกว่าโมเดลตัวแรก ดังนั้นจึงได้เลือกใช้โมเดลตัวที่สอง

5. สรุปอภิปรายผลและข้อเสนอแนะ

5.1 สรุปผลและอภิปรายผล

ในการจัดทำโมเดลการแปลภาษามือด้วยการเรียนรู้เชิงลึก ได้ทำการทดลองสองแบบ ได้แก่ โมเดลตัวแรก ซึ่งใช้ชุดข้อมูลที่เก็บมาจากผู้ใช้เพียง 2 คนมาใช้ในการฝึกฝน และโมเดลตัวที่สอง ซึ่งใช้ชุดข้อมูลที่เก็บมาจากผู้ใช้จำนวน 7 คน พร้อมกับตัดส่วนของ Key points ที่ไม่จำเป็นออก ผลปรากฏว่า ในโมเดลตัวแรกมีค่า Accuracy อยู่ที่ 95% และ ค่า Loss อยู่ที่ 0.2854 แต่มีข้อบกพร่องคือ ในการใช้งานจริง หากทดลองใช้กับผู้ใช้ที่ไม่เคยถูกทำการเก็บข้อมูลฝึกฝนมาก่อน โมเดลจะทำนายค่อนข้างที่จะไม่ถูกต้อง เพราะในการทำท่าทางภาษามือของแต่ละบุคคลนั้น ล้วนแตกต่างกันไป จึงทำให้โมเดลตัวแรก เกิดการ Overfitting เกิดขึ้น และ ในโมเดลตัวที่สอง มีค่า Accuracy อยู่ที่ 93% และค่า Loss อยู่ที่ 0.3692 แต่ในการใช้งานจริง โมเดลตัวที่สองสามารถทำนายผลได้แม่นยำมากกว่าโมเดลตัวแรก แม้จะใช้งานกับผู้ใช้ที่ไม่เคยถูกทำการเก็บข้อมูลฝึกฝนมาก่อน ดังนั้น จึงได้เลือกใช้โมเดลตัวที่สองมาใช้ในเดสก์ท็อปแอปพลิเคชัน เพราะมีประสิทธิภาพในการใช้งานจริงได้ดีและมีความหลากหลายของข้อมูลที่มากกว่าโมเดลตัวแรก

และในการทดลองใช้งานแอปพลิเคชันจริง โดยผู้ใช้ทั้งหมด 7 คน พบว่ามีผลการทำนายถูกต้องเฉลี่ยอยู่ที่ 0.80 ซึ่งสาเหตุเป็นเพราะ ในบางท่าทางนั้น มีตำแหน่ง Key points ที่ไปใกล้เคียงกับท่าทางอื่น ทำให้โมเดลมีการทำนายผลที่ผิด และยังมีในเรื่องของ แสง

เงา และพื้นหลัง ในขณะที่ผู้ใช้แสดงท่าทางเพื่อให้โมเดลทำนายผล ซึ่งมีผลกระทบโดยตรงต่อการทำนายผลของโมเดล เนื่องจากหากมีแสงที่น้อยเกินไปหรือมากเกินไป หรือมีเงามาบดบังขณะที่ผู้ใช้กำลังใช้งานตัวแอปพลิเคชันและพื้นหลังของผู้ใช้ หากมีลวดลายที่มากเกินไป จะทำให้การทำนายผลมีความผิดเพี้ยนเกิดขึ้น นอกจากนี้ในการทำนายผลของตัวแอปพลิเคชัน หากผู้ใช้แสดงท่าทางที่ผู้ใช้ต้องการจะแสดงนั้น ค้างไว้ โดยไม่ขยับไปไหน จะทำให้ตัวแอปพลิเคชันมีการทำนายผลได้ถูกต้องและแม่นยำยิ่งขึ้น เนื่องจากจะเป็นการส่งตำแหน่ง Key points ที่ค่อนข้างมั่นคงเข้าไปให้โมเดลทำการทำนายผล

5.2 ปัญหาและอุปสรรค

1. ปัญหาในการเก็บชุดข้อมูล เนื่องจากต้องใช้ภาษามือของไทย ทางผู้จัดทำจึงต้องถ่ายทำวิดีโอท่าทางภาษามือเอง เนื่องจากภาษามือของไทยไม่ค่อยมีชุดข้อมูลท่าทางตัวอย่างในอินเตอร์เน็ตมากนัก

2. ปัญหาในการฝึกฝนข้อมูล เนื่องจากเกิดการ Overtraining อยู่บ่อยครั้ง จึงได้ทำการทดลองปรับเปลี่ยน Callbacks function และ Batch size อยู่หลายครั้ง

3. ปัญหาในการไม่ได้ทำ Normalize ของชุดข้อมูล ทำให้มีการฝึกฝนข้อมูลที่ค่อนข้างช้า

4. ปัญหาในด้านเวลา เนื่องจากผู้จัดทำพยายามพัฒนาให้โมเดลสามารถใช้งานบนมือถือ แต่ผู้จัดทำมีความรู้และความเข้าใจไม่มากพอ จึงทำให้การดำเนินงานล่าช้า

5. ปัญหาในการใช้งานแอปพลิเคชัน เนื่องจากอัลกอริทึมที่ใช้ในการทำนายของโมเดลมีความซับซ้อนที่ค่อนข้างมาก จึงทำให้แอปพลิเคชันค่อนข้างช้า และ ใช้เวลาค่อนข้างนาน

5.3 ข้อเสนอแนะ

1. ในการเก็บข้อมูล ควรเก็บจากผู้ใช้ที่หลากหลายมากขึ้น

2. ควรทำการ Normalize ชุดข้อมูลก่นที่จะทำการนำไปฝึกฝนข้อมูล

3. ควรพัฒนาในส่วนของท่าทางภาษามือให้มีหลายคำมากขึ้น

4. ควรพัฒนาให้สามารถแปลท่าทางภาษามือออกมาเป็นประโยคได้

5. ควรพัฒนาให้โมเดลสามารถใช้งานบนมือถือได้

เอกสารอ้างอิง

1. สมาคมคนหูหนวกแห่งประเทศไทย (2020). เว็บไซต์ฐานข้อมูลภาษามือไทย. Received: 11 October 2022 from https://www.th-sl.com/?openExternalBrowser=1
2. Manish Nayak (2019). Introduction to the Architecture of Recurrent Neural Networks (RNNs). Received: 11 October 2022 from https://pub.towardsai.net/introduction-to-the-architecture-of-recurrent-neural-networks-rnns-a277007984b7
3. Praveen Kumar Anwla (2019). Recurrent Neural Network (RNN) architecture explained in detail. Received: 11 October 2022 from https://towardsmachinelearning.org/recurrent-neural-network-architecture-explained-in-detail/?fbclid=IwAR0SsNWUBvxyFkOuup4OA_502WpFGUMIlZjavdaCIJgejwfjKqWLX6gqmfA

4. Afshine Amidi, Shervine Amidi (2020). Recurrent Neural Networks cheatsheet. Received: 11 October 2022 from https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
5. Surapong Kanoktipsatharporn (2019). Recurrent Neural Network (RNN) คืออะไร Gated Recurrent Unit (GRU) คืออะไร สอนสร้าง RNN ถึง GRU ด้วยภาษา Python. Received: 11 October 2022 from https://www.bualabs.com/archives/3103/what-is-rnn-recurrent-neural-network-what-is-gru-gated-recurrent-unit-teach-how-to-build-rnn-gru-with-python-nlp-ep-9/?fbclid=IwAR1PTy5gWjP8LksmrPkBJNFOg05LMUInXHI9uQHj6zZ_yLLkW1a2295qr2A
6. Ankita Wadhawan, Parteek Kumar (2020). Deep learning-based sign language recognition system for static signs. Received: 11 October 2022 from Deep learning-based sign language recognition system for static signs | SpringerLink
7. Kavana KM,Suma NR mediapipe (2022) Received: 22 March 2023 from https://www.irjmets.com/uploadedfiles/paper//issue_6_june_2022/27011/final/fin_irjmets1656344520.pdf?fbclid=IwAR3d8nfQ_qNYbdfa8yw_uheXeqBguPBzc9P-vr8InXzb25j9xoOBrILMhQ8
8. Colah (2015) Understanding LSTM Networks Received: 22 March 2023 from http://colah.github.io/posts/2015-08-Understanding-LSTMs/?fbclid=IwAR3LjSNWe79gu_6biHTt7ANsa3JEJMg3vu7LQWSYlWyNhQzlYmuJBORwbXE
9. Yugesh Verma(2021) A Complete Understanding of Dense Layers in Neural Networks Received: 22 March 2023 from https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/?fbclid=IwAR1_KeaiKHAoXY_niadcN9j813m6XoN2gduPWRfwGNObV7NQ5mjogUie_M
10. SAGAR SHARMA(2017) Activation Functions in Neural Networks Received: 22 March 2023 from https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
11. Daniel Johnson (2023) Confusion Matrix in Machine Learning with EXAMPLE Networks Received: 22 March 2023 from https://www.guru99.com/confusion-matrix-machine-learning-example.html#:~:text=A%20confusion%20matrix%20is%20a%20performance%20measurement%20technique,its%20related%20terminology%20can%20be%20a%20little%20confusing