



AI for Healthcare

แอปพลิเคชันจำแนกโรคมะเร็งผิวหนังที่นิยมในไทย

จัดทำโดย

ณรีพัฒน์ รุ่งรำพรรณ	6509650369 (ผู้ประสานงาน)
รังสิมันต์ ไหวติง	6309650627
ปาณิสรา เนาวรัตน์	6509650039
ภัทรวัต ทานะมัย	6509650153
ธัชพัฒน์ วิลาราช	6509650443
นันทน์ภัส เพ็ชรทอง	6509650518
ศักดิ์ศิรินภา เตมศักดิ์	6509650724
อชิรญาณ ชูเชิด	6509650765

เสนอ

ผศ.ดร ธนาธร ทะนานทอง

รายงานนี้เป็นส่วนหนึ่งของรายวิชา คพ.265 Artificial Intelligence fundamentals

คณะวิทยาศาสตร์และเทคโนโลยี สาขาวิทยาการคอมพิวเตอร์

ภาคการเรียนรู้ที่ 2 ปีการศึกษา 2566

คำนำ

ด้วยความก้าวหน้าของเทคโนโลยี AI ที่มีบทบาทสำคัญในการปฏิวัติวงการสุขภาพ รายงานนี้จึงถูกจัดทำขึ้นเพื่อสำรวจและวิเคราะห์การใช้งาน AI ในด้านการดูแลสุขภาพ ซึ่งเป็นส่วนหนึ่งของรายวิชา คพ.265 พื้นฐานของปัญญาประดิษฐ์ คณะวิทยาศาสตร์และเทคโนโลยี สาขาวิทยาการคอมพิวเตอร์ ภาคการเรียนรู้ 2 ปีการศึกษา 2567 ผ่านการศึกษาครั้งนี้ ผู้จัดทำหวังว่าจะสามารถนำเสนอมุมมองใหม่ๆ และความเข้าใจที่ลึกซึ้งยิ่งขึ้นเกี่ยวกับการประยุกต์ใช้ AI ในการสนับสนุนการตัดสินใจทางการแพทย์ การวินิจฉัยโรค และการดูแลผู้ป่วย ซึ่งจะนำไปสู่การพัฒนาคุณภาพชีวิตและการดูแลสุขภาพที่ดียิ่งขึ้น

ขอขอบคุณอาจารย์ผู้สอน ผู้เชี่ยวชาญ และเพื่อนๆ ที่ได้ให้คำแนะนำและสนับสนุนในการจัดทำรายงานนี้ และหวังว่าผลงานนี้จะเป็นประโยชน์ต่อผู้ที่สนใจในการใช้ AI เพื่อการดูแลสุขภาพ

ลงชื่อ (กลุ่ม ไก่หมดจรรย์)

วันที่ 3 พฤษภาคม 2567

สารบัญ

เรื่อง	หน้า
คำนำ.....	ก
สารบัญ.....	ข
บทที่ 1 บทนำ.....	1
1.1 ที่มาและความสำคัญ.....	1
1.2 วัตถุประสงค์.....	1
1.3 ประโยชน์ของโครงการ.....	1
บทที่ 2 งานวิจัยที่เกี่ยวข้อง/ตัวอย่างแอปพลิเคชัน.....	3
2.1 มะเร็งผิวหนัง.....	3
2.2 การจำแนกรูปภาพ.....	3
2.3 การเรียนรู้เชิงลึก.....	3
2.4 โครงข่ายประสาทเทียมคอนโวลูชัน.....	3
2.5 งานวิจัยที่เกี่ยวข้อง.....	4
2.6 แอปพลิเคชัน.....	7
บทที่ 3 วิธีการดำเนินการโครงการ.....	10
3.1 การเก็บข้อมูล/การเตรียมข้อมูล.....	10
3.2 การสร้างโมเดล.....	12
3.3 คุณสมบัติของแต่ละโมเดล.....	13
3.4 การทำงานของโมเดล.....	14
3.5 วัดประสิทธิภาพของโมเดล.....	15
3.6 การปรับพารามิเตอร์.....	16
3.7 เขียนอธิบาย source code.....	17
3.8 การออกแบบ UX/UI ของ Application.....	37
บทที่ 4 ผลลัพธ์การดำเนินการ/อภิปรายผล.....	42
บทสรุปและแนวทางแนะนำ.....	47
อุปสรรคและปัญหาที่พบ.....	47
บรรณานุกรม.....	48
ภาพผนวก ก	
เครื่องมือที่ใช้.....	50

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ข้อมูลจาก thainakarin.co.th บอกว่าจากสถิติข้อมูลมะเร็งประเทศไทยระหว่างปี พ.ศ. 2559 – 2561 โดย สถาบันมะเร็งแห่งชาติ พบผู้ป่วยมะเร็งผิวหนังรายใหม่เฉลี่ยถึง 4,374 คนต่อปี หรือวันละ 12 คน โดยพบมากในผู้หญิงมากกว่าผู้ชาย สาเหตุหลักที่สำคัญมาจากการโดนแสงแดด ซึ่งมะเร็งผิวหนังบางชนิดมีความรุนแรงมากและอาจนำไปสู่การเสียชีวิตได้หากไม่ได้รับการรักษาที่เหมาะสม โดยมะเร็งผิวหนังที่มีโอกาสพบมากในไทยมีทั้งหมด 3 ชนิด ได้แก่

1. Basal cell carcinoma (BCC) : พบบ่อยที่สุด กว่า 80% ของผู้ป่วย มักปรากฏบนผิวหนังที่โดนแดดบ่อย ๆ ลักษณะเป็นตุ่มสีเข้มน้ำวาวคล้ายไข่มุก อาจมีเลือดซึมออกมา หรือเป็นก้อนแข็ง ก้อนเนื้อจากมะเร็งชนิดนี้จะโตช้า และจะโตไปเรื่อย ๆ จนอาจแผลแตกในที่สุด ทำให้มีเลือดออกและกลายเป็นแผลเรื้อรัง หากไม่ได้รับการรักษาอาจลุกลามและทำให้เกิดการเสียชีวิต
2. Squamous cell carcinoma (SCC) : พบมากเป็นอันดับที่ 2 มีลักษณะเป็นก้อนเนื้อแข็งหรือแผลที่มีขอบเป็นขุย เลือดออกง่าย และแผลจะค่อย ๆ ขยายขนาดไปเรื่อย ๆ และกลายเป็นแผลเรื้อรังในที่สุด หากไม่ได้รับการรักษาอาจลุกลามไปยังอวัยวะอื่น
3. Melanoma : เป็นมะเร็งผิวหนังที่รุนแรงที่สุด เริ่มแรกจะส่งผลให้เกิดจุดบนผิวหนัง ดูกคล้ายไฟที่มีลักษณะผิดปกติและมีขนาดใหญ่ขึ้นอย่างรวดเร็ว มีขอบเขตไม่เรียบและ อาจมีสีไม่สม่ำเสมอ ทั้งนี้ที่บริเวณแผลอาจตกสะเก็ดหรือมีอาการเลือดออกด้วยเช่นกัน หากไม่ได้รับการรักษาอาจลุกลามไปยังอวัยวะอื่นและเป็นอันตรายถึงชีวิต โดยเป็นสาเหตุการตายถึง 75% ของผู้ป่วยมะเร็งผิวหนังทั้งหมด

1.2 วัตถุประสงค์

1. เพื่อพัฒนาเทคโนโลยีใหม่ในการตรวจจับมะเร็งผิวหนังที่มีประสิทธิภาพสูงขึ้นกว่าเดิมโดยใช้ปัญญาประดิษฐ์ (AI)
2. เพื่อช่วยให้การวินิจฉัยมะเร็งผิวหนังเป็นไปอย่างรวดเร็วและแม่นยำยิ่งขึ้น
3. เพื่อช่วยลดภาระงานของแพทย์และช่วยให้ผู้ป่วยได้รับการวินิจฉัยและการรักษาที่เหมาะสมอย่างรวดเร็ว

1.3 ประโยชน์ของโครงการ

1. ทำให้การตรวจมะเร็งผิวหนังเป็นไปได้ง่ายและสะดวกมากขึ้น โดยไม่จำเป็นต้องเข้ารับการรักษาที่โรงพยาบาลเสมอไป
2. ช่วยให้แพทย์สามารถตรวจจับมะเร็งผิวหนังได้ในระยะเริ่มต้น ซึ่งเป็นช่วงที่การรักษามีโอกาสสำเร็จสูงสุด

3. เพิ่มความตระหนักรู้แก่ประชาชนทั่วไปเกี่ยวกับมะเร็งผิวหนังและการป้องกัน ทำให้สามารถลดอัตราการเกิดมะเร็งผิวหนังได้

บทที่ 2

งานวิจัยที่เกี่ยวข้อง/ตัวอย่างแอปพลิเคชัน

2.1 มะเร็งผิวหนัง

มะเร็งผิวหนังเป็นปัญหาสุขภาพที่สำคัญในปัจจุบันถือเป็นอุบัติการณ์ผู้ป่วยโรคมะเร็งผิวหนังที่ทั่วโลกพบซึ่งเป็นผลเกิดจากการเจริญเติบโตของเซลล์ผิวหนังที่ผิดปกติที่มีลักษณะของเนื้อร้ายที่เกิดขึ้นบนผิวหนังและเยื่อหุ้มหากไม่ได้รับการรักษาโดยทันตแพทย์ที่อาจทำให้เป็นอันตรายถึงแก่ชีวิตได้แต่ด้วยการวินิจฉัยโรคมะเร็งผิวหนังในระยะเริ่มต้นนั้นจะสามารถช่วยลดโอกาสการลุกลามและลดอัตราการเสียชีวิตของผู้ป่วยลงได้ทว่าในการวินิจฉัยมะเร็งผิวหนังจำเป็นต้องอย่างมากจะต้องอาศัยแพทย์ผู้เชี่ยวชาญในการวินิจฉัยเท่านั้น

2.2 การจำแนกรูปภาพ

การจำแนกรูปภาพ (Image Classification) เป็นการจัดกลุ่มหรือประเภทให้กับรูปภาพด้วยกระบวนการเรียนรู้เชิงลึก โดยเป็นการนำข้อมูลรูปภาพมาทำการสกัดคุณลักษณะพิเศษ (Feature) ของวัตถุภายในรูปภาพ เช่น แสง สี เส้นขอบ และความโค้ง โดยคุณลักษณะพิเศษดังกล่าวจะถูกใช้ในการเรียนรู้ของแบบจำลองเพื่อนำไปทำการจำแนกรูปภาพ โดยเป็นการจำแนกให้อยู่ในประเภท (Class) ที่มีคุณลักษณะพิเศษใกล้เคียงกัน อีกทั้งมีนักวิจัยหลายคนนำเทคนิคการจำแนกรูปภาพมาประยุกต์ใช้ร่วมกับการตรวจหาโรค และวินิจฉัยโรคมะเร็งผิวหนัง เช่น ขวัญกมลดิฐกัญจน์, ปิยะวัฒน์ หนูเล็ก และ กรวิทย์ พฤษชัยนิมิต ที่นำเสนอการวินิจฉัยมะเร็งผิวหนังจากภาพถ่ายโทรศัพท์ร่วมกับโครงข่ายประสาทเทียมสังวัตนาการหรือCNN โดยมีการจำแนกภาพ ประกอบด้วยภาพโรคมะเร็งผิวหนัง 3 ชนิดและโรคผิวหนัง 3 ชนิด

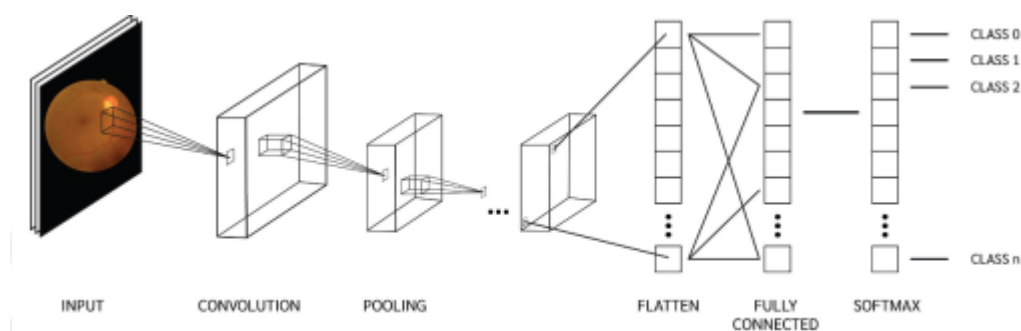
2.3 การเรียนรู้เชิงลึก (Deep Learning)

การเรียนรู้เชิงลึกเป็นความสามารถในการใช้หลักการคิดพร้อมทั้งหลักการประมวลผลแบบสมองของมนุษย์ซึ่งเรียกว่าโครงข่ายประสาทเทียม (Artificial Neural Network) ซึ่งได้ถูกสร้างขึ้นโดยการนำระบบโครงข่ายประสาทเทียมมาทำการซ้อนกันจำนวนหลายชั้นพร้อมทั้งเรียนรู้ข้อมูลโดยการนำชุดข้อมูลสำหรับการเรียนรู้ของแบบจำลองเพิ่มเข้าไปในส่วนของชั้นรับข้อมูล(Input Layer) เพื่อส่งต่อข้อมูลไปประมวลผลที่ชั้นซ่อน (Hidden Layer) และนำเสนอผลลัพธ์ของการประมวลผลที่ชั้นแสดงผล (Output Layer)

2.4 โครงข่ายประสาทเทียมคอนโวลูชัน (Convolutional Neural Network)

โครงข่ายประสาทเทียมคอนโวลูชัน (Convolutional Neural Network: CNN) เป็นหนึ่งในการเรียนรู้เชิงลึกแบบมีผู้ฝึกสอน (Supervised Learning) ซึ่งถูกนำไปประยุกต์ใช้กันอย่างแพร่หลาย รวมไปถึงการประยุกต์ใช้กับการประมวลผลภาพและข้อมูลที่เป็นลำดับ เป็นวิธีการที่ใช้ในการแก้ปัญหาทางด้านการรับรู้ภาพ เช่น การ

จำแนกประเภทภาพ, การตรวจจับวัตถุ เป็นต้น โครงสร้างพื้นฐานของ CNN ประกอบด้วยชั้นต่าง ๆ และโครงสร้างทั่วไปของ CNN จะแสดงดังภาพ



2.5 งานวิจัยที่เกี่ยวข้อง

ในปีที่ผ่านมา มีนักวิจัยหลายคนได้นำเสนองานวิจัยที่ใช้เทคนิคการรู้จำรูปภาพเข้ามาช่วยในการคัดกรองโรคมะเร็งผิวหนังในเบื้องต้นสำหรับการแยกแยะผิวหนังที่เป็นมะเร็งผิวหนังกับผิวหนังที่ไม่เป็นโรคมะเร็งผิวหนังซึ่งในกระบวนการนี้ไม่ต้องผ่านการวินิจฉัยจากแพทย์ผิวหนังโดยตรงส่งผลให้การคัดกรองโรคเป็นไปด้วยความรวดเร็วและอาจลดการสูญเสียโรคมะเร็งผิวหนังของผู้ป่วยรวมทั้งลดภาระงานของแพทย์ผิวหนังอีกด้วย

จากปัญหาที่ได้กล่าวมา Surapong Kanoktipsatharporn ได้นำเสนอการสร้าง AI ด้วยวิธี Deep learning โดยวินิจฉัยโรคมะเร็งผิวหนัง 7 ชนิด โดยใช้ชุดข้อมูล HAM10000 (Human Against Machine with 10000 training images) ซึ่งประกอบด้วยชุดข้อมูลโรคผิวหนัง ของผิวหนังที่มีจุดต่างและย้อมสีต่าง ๆ จำนวน 10,015 รูป ประกอบด้วย Melanocytic nevi (ไฝเมลานโนไซต์) มีจำนวน 6705 รูป, Melanoma (มะเร็งเมลานโนมา) มีจำนวน 1113 รูป, Benign keratosis (กระเนื้อ) มีจำนวน 1099 รูป, Basal cell carcinoma (มะเร็งเบเซลเซลล์) มีจำนวน 514 รูป, Actinic keratoses (ผื่นแฉกทีนิกเคอราโทซิส) จำนวน 327 รูป, Vascular lesions (เนื้องอกของหลอดเลือด) มีจำนวน 142 รูป และ Dermatofibroma (เนื้องอกของเส้นใยในผิวหนัง) มีจำนวน 115 รูป โดยงานวิจัยนี้ใช้โมเดล ResNet50 เวอร์ชัน fastai : XResnet50 ซึ่งเป็นโมเดลที่มีการปรับปรุงและปรับแต่งเพื่อให้มีประสิทธิภาพและความเร็วในการ train ที่มีประสิทธิภาพมากกว่าเดิม XResNet50 มีการใช้โครงสร้างที่ลดลงและความกว้างเพิ่มขึ้น โดยโมเดล XResNet50 มีการลดลงของจำนวน depth และเพิ่มจำนวนของชั้นและความกว้างของชั้น width ซึ่งช่วยเพิ่มประสิทธิภาพในการ train และในเวลาเดียวกันก็ช่วยลดปัญหาการเกิด overfitting ด้วย ทำให้โมเดลมีความสามารถในการแยกแยะประเภทของภาพได้อย่างแม่นยำและมีประสิทธิภาพสูงในการทำนายประเภทของภาพและสามารถใช้งานได้หลากหลายมุ่งเน้นเพื่อการประมวลผลภาพเพื่อจำแนกว่าเป็นเนื้อเยื่อที่เป็นมะเร็งหรือไม่ โดยใช้วิธีการสร้างแบบจำลอง (Model) ที่ฝึกสอนด้วยภาพจากฐานข้อมูลที่มีการประเมินความเสี่ยงของโรคแล้วเพื่อให้มีความแม่นยำในการวินิจฉัย ผลการทดสอบหลังจาก Progressive Resizing ทำให้ได้ Accuracy เท่ากับ 94.5%

ขวัญกมลดิษฐกัญจน์, ปิยะวัฒน์ หนูเล็ก และ กรวิทย์ พฤษชัยนิมิต นำวินิจฉัยมะเร็งผิวหนังจากภาพถ่าย

โทรศัพท์มาพัฒนาร่วมกับโครงข่ายประสาทเทียมสังวัตนาการหรือ CNN เพื่อทำการเรียนรู้จากชุดข้อมูลภาพของผู้ป่วยที่มีโรคมะเร็งผิวหนังและภาพของผู้ป่วยที่ไม่มีโรค จากนั้นใช้โมเดลที่เรียนรู้แล้วเพื่อทำนายหรือวินิจฉัยว่าภาพถ่ายใหม่ที่เข้ามานั้นมีโรคมะเร็งผิวหนังหรือไม่ โดยชุดข้อมูลที่ใช้เป็นชุดข้อมูล PAD-UFES-20 จากสถาบัน Federal University of Espirito Santo (UFES) ซึ่งมีการจำแนกภาพ ประกอบด้วยภาพโรคมะเร็งผิวหนัง 3 ชนิดและโรคผิวหนัง 3 ชนิด ในงานวิจัยนี้ใช้สถาปัตยกรรม CNN 5 รูปแบบ ในการวินิจฉัยเพื่อหาประสิทธิภาพที่ดีที่สุดในการใช้โมเดล ประกอบด้วยโมเดล VGG19 นำมาใช้เนื่องจากมีโครงสร้างที่เรียบง่ายเข้าใจง่ายและมีการแทนที่ Hyperparameter จำนวนมากโดยเครือข่าย VGG จะใช้ Multiple Convolutional Layers เน้นไปที่การออกแบบ Layer โดยใช้ตัวกรองขนาด 3×3 (Filters), EfficientNetB7 นำมาใช้เพราะมีเทคนิค Compound Scaling ซึ่งจะเป็นการเพิ่มทั้ง ความกว้าง (Width), ความลึก (Depth) และความละเอียด (Resolution) ด้วยอัตราส่วนคงที่ (Constant Ratio), Xception นำมาใช้เนื่องจากมีความสามารถช่วยปรับปรุงโมดูลและสถาปัตยกรรมการเริ่มต้นด้วยสถาปัตยกรรมที่เรียบง่ายและประสิทธิภาพเท่ากับ ResNet และ InceptionV4 แต่มีการทำงานที่น้อยลงด้วยรูปแบบและรหัสที่ง่ายกว่า, DenseNet121 นำมาใช้เนื่องจากถูกพัฒนามาจาก ResNet การทำงานมีความคล้ายคลึงกันแต่แตกต่างเพียงการนำส่งข้อมูลในแต่ละชั้น และ ResNet152V2 นำมาใช้เนื่องจากเป็นการแก้ปัญหาเรื่อง Vanishing Gradient ซึ่งจะเกิดขึ้นกับโครงข่ายที่มีความลึกที่ค่อนข้างมาก โดยใช้เทคนิคการออกแบบ Module ที่มีลักษณะเป็นทางลัดลงในเครือข่าย ผลของการทดสอบโมเดลเป็นดังนี้ EfficientNetB7 มีค่าความแม่นยำที่ 49%, Xception ค่าความแม่นยำที่ 72.5%, ResNet152V2 มีค่าความแม่นยำที่ 76.5%, DenseNet121 ค่าความแม่นยำที่ 78.5% และ VGG19 ค่าความแม่นยำที่ 64% ซึ่งโมเดลที่มีความแม่นยำสูงสุดที่ทดสอบคือ DenseNet121 จึงนำโมเดลนี้มาใช้เป็นหลักในการวินิจฉัยโรคมะเร็งผิวหนังของงานวิจัยเนื่องจากมีความแม่นยำสูงสุด โดยเมื่อนำโมเดลมาทำการทดสอบพร้อมปรับค่าพารามิเตอร์ ค่าความแม่นยำมากที่สุดที่ 81.50% ที่ขนาด Batch Size เท่ากับ 1 และมีความผิดพลาดของโมเดลที่ 18.5% ทั้งนี้โมเดลยังมีความแม่นยำต่ำเนื่องจากจำนวนรูปภาพที่มีอย่างจำกัดและระยะเวลาที่โมเดลไม่สามารถเรียนรู้จากชุดข้อมูลตั้งต้นได้อย่างมีประสิทธิภาพ เนื่องจากกระบวนการมีความซับซ้อนและ dataset ขนาดใหญ่ทำให้เวลาประมวลผลมีระยะเวลานาน

Natratanon Kanraweekultana ได้พบว่าในงานวิจัยนี้อาจเกิดปัญหานึงขึ้นมาได้นั้นคือ Overfitting คือปัญหาที่เกิดจากการที่ใช้ Training Data set มีค่าความถูกต้องในการบ่งบอกคลาสเป้าหมายสูง แต่เมื่อนำไปใช้กับข้อมูล Test Data set กลับได้ค่าความถูกต้องต่ำ หรือ หรือก็คือสามารถเรียนรู้ข้อมูลจาก Training Data set ได้ดีมาก แต่เมื่อนำไปใช้กับข้อมูลที่ไม่เคยพบมาก่อนกลับทำไม่ได้มีดี โดยการแก้ปัญหา Overfitting ก็คือการลดความซับซ้อนในการปรับ Parameter โดยเนื่องจากโหนดมันมีอิทธิพลมากไป จึงทำการสุ่มตัดบางโหนดออกไป เพื่อลดความซับซ้อนที่มากเกินไปที่จะก่อให้เกิดการ Overfitting แต่หากลดความซับซ้อนในการปรับค่าน้อยจนเกินไป ก็จะทำให้เกิดปัญหา Under-fitting แต่ถ้า ปรับค่าน้อยจนเกินไป ก็จะทำให้เกิดปัญหา Over-fitting ได้ ดังนั้นในการปรับค่าของ Parameter ควรทำให้เหมาะสม เพื่อไม่ให้เกิดปัญหาใดๆ โดยผู้ทดลองได้ทดลองใช้ชุดข้อมูลภาพ Data set fashion mnist ซึ่งเป็นชุดข้อมูลเก็บรูปภาพชุดเสื้อผ้า โดยเริ่มต้นจากการ Train data และ Test data เพื่อดูการกระจายตัวของข้อมูลในแต่ละ Class จากนั้นสุ่มภาพออกมาและแบ่งข้อมูลเป็นส่วน

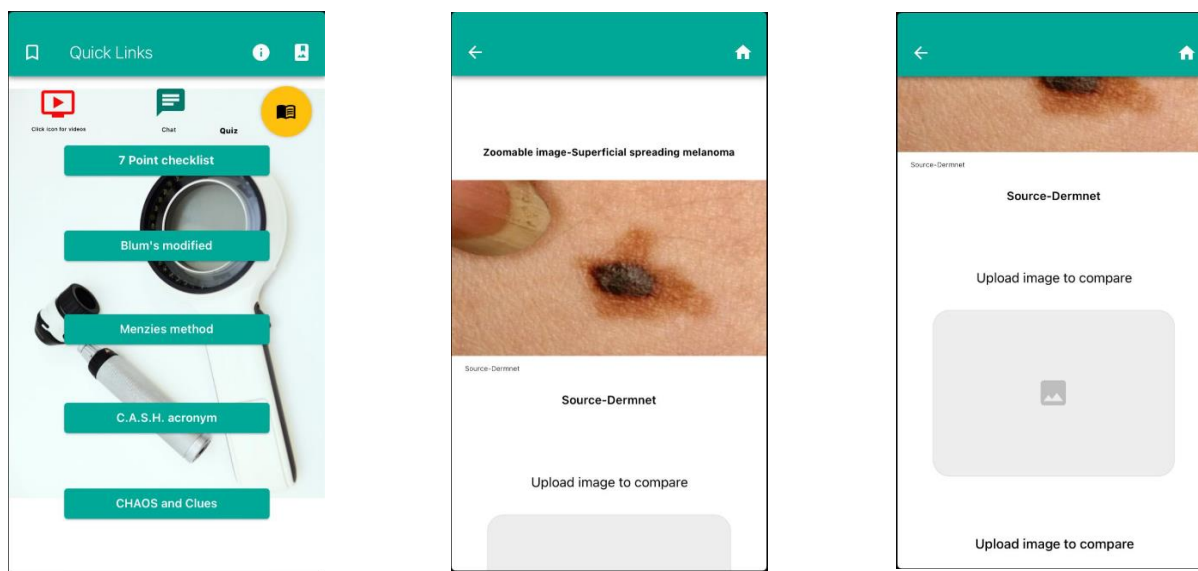
ของ train และ test จากนั้นสร้างโมเดลโดยภาพที่สุ่มออกมาทั้งหมด 32 แผ่น แผ่นละ 3×3 ใช้ `model.add(MaxPooling2D((2, 2)))` เพื่อลดขนาดของภาพให้ขนาดเล็กลง และใช้ `kernel_size=(3, 3)` เพื่อกำหนดขนาดภาพ จากนั้นสร้างโมเดลเพื่อทำ Classification และเมื่อทำการนับจำนวน Parameter แล้วพบว่าทั้งหมด 241,546 Parameter และเมื่อทำการ Train model แล้วพบว่าเกิดปัญหา Overfitting จึงจำเป็นต้องแก้ปัญหาด้วยการ Dropout ลดจำนวนโหนด ด้วยการแก้ไขโมเดลด้วยการเพิ่มการ Dropout ในแต่ละชั้นโดยทำการตัดออก 4 จุดที่ 25%, 25%, 40% และ 30% จากนั้นเมื่อทำการ Dropout ก็พบได้ว่าสามารถแก้ปัญหา Overfitting โดยค่า Validation loss ลดลงเรื่อยๆ ถึงแม้จะทำให้ค่า Accuracy ลดลงก็ตาม

ภัทรมน พันธุ์แพง และ สัญญา พันธุ์แพง ได้ทำการวิจัยแอปพลิเคชันวิเคราะห์ภาพถ่ายจากอุปกรณ์ตรวจเท้าสำหรับผู้ป่วยเบาหวาน ผู้ใช้สามารถดำเนินการวิเคราะห์ภาพถ่ายได้ด้วยตัวเองจากการถ่ายภาพและทำแบบประเมิน โดยแอปพลิเคชันที่ถูกพัฒนาขึ้นจะอยู่ในรูปแบบของ Native Application บน แอนดรอยด์ โดยอาศัย Library หรือ SDK ของแพลตฟอร์มนั้นๆ และพัฒนาด้วยภาษาจาวา งานวิจัยได้มีการศึกษาแบบประเมินเท้าจากคู่มือแนวทางการดูแลเท้าในผู้ป่วยเบาหวานในชุมชน เพื่อนำไปแสดงเป็นรายงานการวิเคราะห์ภาพถ่ายเท้า จากนั้นได้มีการใช้ Application Programming Interface หรือ API เพื่อแลกเปลี่ยนข้อมูลจากระบบหนึ่งไปสู่ระบบอื่นๆ หลังจากนั้นได้มีการนำไปทดสอบกับผู้ป่วยเบาหวานจำนวน 10 คน ที่พักอาศัยในตำบลหมอกจำแป๋ อำเภอเมืองแม่ฮ่องสอน จังหวัดแม่ฮ่องสอน และผู้เชี่ยวชาญ โดยจากการประเมินของผู้เชี่ยวชาญพบว่าแอปพลิเคชันมีความถูกต้อง น่าเชื่อถือ และใช้งานได้ง่าย และจากผู้ป่วย มีการให้แบบประเมินพบว่าผู้ป่วยมีการประเมินว่ามีระดับความพอใจที่ พอใจมาก

พิชาดา สายเชื้อ ได้ทำการวิจัยการจำแนกรูปภาพจอประสาทตาที่มีภาวะเบาหวานขึ้นตาด้วยการเรียนรู้เชิงลึก งานวิจัยนี้ได้ใช้ชุดข้อมูลรูปภาพจำนวน 3 ชุด ได้แก่ APTOS 2019 Blindness Detection , Diabetic Retinopathy Detection และ Diabetic Retinopathy Dataset ซึ่งเป็นข้อมูลรูปภาพที่แต่ละภาพแสดงเฉพาะบริเวณของจอประสาทตาเท่านั้น ประกอบไปด้วย 2 คลาส ได้แก่ จอประสาทตาปกติ และจอประสาทตาที่มีภาวะเบาหวานขึ้นตา แต่เนื่องจากภาพที่นำมาวิจัยมีขนาดไม่เท่ากันจึงมีการกำหนดขนาดภาพให้เท่ากันที่ 299×299 Pixels โดยมีชนิดรูปภาพเป็น JPEG โดยงานวิจัยนี้มีการใช้ CNN โมเดล 5 รูปแบบได้แก่ ResNet50, ResNet50V2, Xception, IncetionV3 และ DenseNet121 ร่วมกับเทคนิค 4 เทคนิคเริ่มจาก Fine-tuning Hyperparameter เมื่อปรับแต่ง Hyperparameter Optimization และ Learning Rate ให้เหมาะสม จากแบบจำลอง Xception ได้อัตราจำแนกรูปภาพสูงสุดอยู่ที่ร้อยละ 84.07 เทคนิคต่อมา Data Augmentation นั้นคือการนำภาพที่มีอยู่มาทำการเปลี่ยนแปลงมุมมองโดยมีการทำ 3 รูปแบบนั้นคือ Rotation Flip และ Zoom จากแบบจำลอง Xception ได้อัตราจำแนกรูปภาพสูงสุดอยู่ที่ร้อยละ 85.50 เทคนิคต่อมา Ensemble CNN นั้นคือการนำการเรียนรู้จากข้อมูลชุดเดียวกันมาทำการ Prediction เพื่อให้ได้ค่า Probability ของแต่ละแบบจำลอง และนำค่าที่ได้จากทั้ง 2 แบบจำลอง ไปทำการ Vote เพื่อหาค่าเฉลี่ยของ คำตอบที่ถูกต้องที่สุด โดยเทคนิคนี้จากแบบจำลอง Xception ได้อัตราจำแนกรูปภาพสูงสุดอยู่ที่ร้อยละ 86.11 เทคนิคสุดท้าย Fusion CNN คือการดึงเอาคุณสมบัติในการจำแนกคุณลักษณะพิเศษของแต่ละ CNN มา รวมเข้าด้วยกัน และ

ทำการสร้างเป็นแบบจำลองรูปแบบใหม่ที่มีความสามารถในการจำแนกประเภท ของวัตถุในรูปภาพของทุก CNN โดยมีการเพิ่ม Dense Layer และ Dropout Layer เพื่อลดการเกิดปัญหา Overfitting เมื่อทำการ ทดสอบจากแบบจำลอง Xception ได้อัตราจำแนกรูปภาพสูงสุดอยู่ที่ร้อยละ 86.30 ซึ่งสูงที่สุดในงานวิจัยนี้

2.6 ตัวอย่างแอปพลิเคชัน



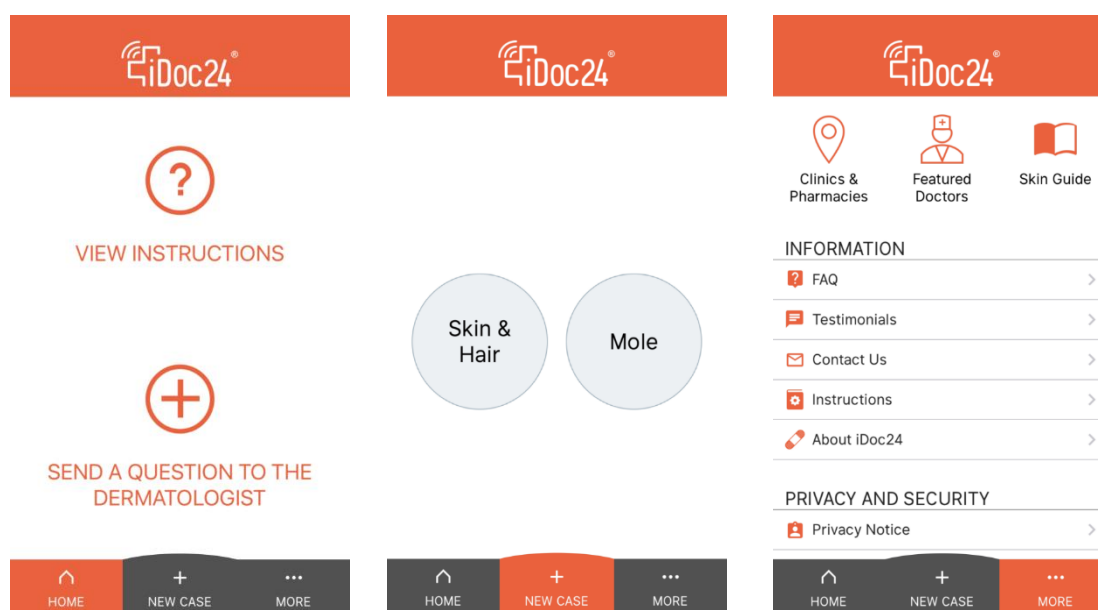
2.6.1 แอปพลิเคชัน Skin cancer scoring

เป็นแอปที่ไว้ให้คะแนนรูปภาพที่เรานำมาเปรียบเทียบกับความใกล้เคียงกับรูปภาพที่เป็นโรคมะเร็งรี เปลา

ความสามารถ :

- มีปุ่มให้กดถ่ายรูปหรือเพิ่มรูป
- สามารถแจ้งปัญหาได้
- สามารถให้ข้อมูลเกี่ยวกับโรคมะเร็งได้
- สามารถดูประวัติการวินิจฉัย
- มีความง่ายของภาษา

ส่วนการนำมาปรับใช้ ในด้านของการเพิ่มในส่วนของการสามารถแชทถามกับแพทย์ได้โดยตรง



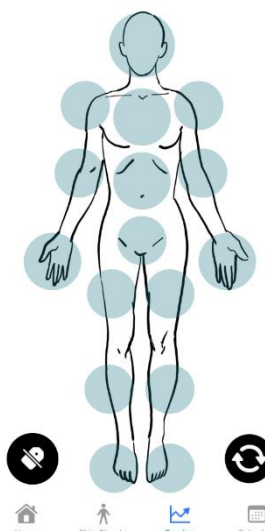
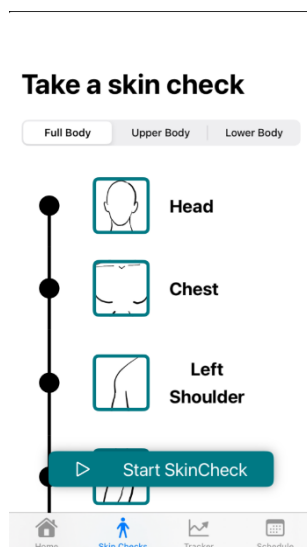
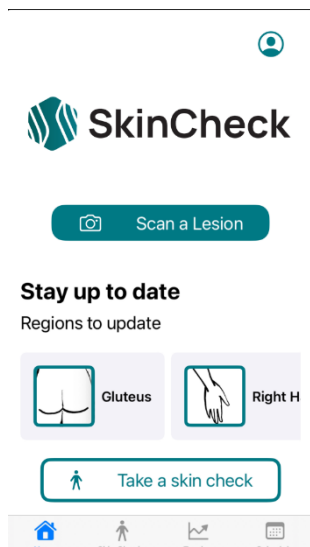
2.6.2 แอปพลิเคชัน idocs24

หน้า ux/ui ใช้งานง่ายสามารถส่งคำถามให้แพทย์ผิวหนังได้โดยตรงสามารถดูlocation คลินิกผิวหนังได้ มีคำแนะนำเกี่ยวกับผิวหนัง

ความสามารถ :

- มีปุ่มให้กดถ่ายรูปหรือเพิ่มรูป
- สามารถแจ้งปัญหาได้
- มีความง่ายของภาษา

ส่วนการปรับใช้ deeskin สามารถนำส่วนของการติดต่อแพทย์ผิวหนังได้เลยมาปรับใช้ในกรณีที่มีปัญหาทางผิวหนังเกิดขึ้นทางแพทย์จะได้ตรวจสอบวินิจัยได้ต่อไปได้ในทันทีและสามารถนำส่วนของlocation ของ clinic มาใช้จะได้สะดวกต่อการไปพบแพทย์ผิวหนัง



2.6.3 แอปพลิเคชัน Skin Check

เป็นแอปที่ให้ผู้ใช้กรอกข้อมูลส่วนต่างๆของร่างกายและแนบรูปภาพประกอบ ซึ่งมีฟังก์ชันการทำงานครบ สามารถวินิจฉัยผิวหนังแต่ละส่วนของร่างกายได้ อาจมีการใช้งานลำบากเพราะต้องเลือกในแต่ละส่วนของร่างกาย

ความสามารถ :

- มีปุ่มให้กดถ่ายรูปหรือเพิ่มรูป
- สามารถให้ข้อมูลเกี่ยวกับโรคมะเร็งได้
- มีตำแหน่งของโรคให้เลือก
- สามารถดูประวัติการวินิจฉัย
- มีความง่ายของภาษา

สิ่งที่สามารถนำมาปรับใช้ได้ในอนาคตคือ การเพิ่มในส่วนการระบุตำแหน่งที่ต้องการจะวินิจฉัยเพื่อเพิ่มความสามารถในการวินิจฉัยให้แม่นยำขึ้น

บทที่ 3

วิธีการดำเนินการโครงการ

3.1 การเก็บข้อมูล/การเตรียมข้อมูล

3.1.1 การเก็บข้อมูล

ใช้จาก dataset ที่มีอยู่แล้วจาก Kaggle (20 Skin Diseases Dataset) โดยนำมาแบ่งเป็นชุดข้อมูล ออกเป็น train, test และ valid โดยจัดกลุ่ม dataset เพื่อการฝึกฝนสำหรับ AI ในแต่ละชุดข้อมูลออกเป็น 4 กลุ่มตามลักษณะโรคผิวหนังที่คนไทยนิยมเป็นสูงสุด 3 อันดับแรกได้แก่ Basal cell carcinoma, Squamous cell carcinoma, และ Melanoma ตามลำดับ ส่วนอีกหนึ่งลักษณะเป็นผิวหนังปกติคือ คือ Normal skin โดยรายละเอียดข้อมูลของแต่ละ dataset มีดังนี้

1. Basal cell carcinoma เป็นมะเร็งผิวหนังที่พบบ่อยที่สุด กว่า 80% ของผู้ป่วยมะเร็งผิวหนัง (อันดับ 1) จะเป็นมะเร็งชนิดนี้ มักปรากฏบนผิวหนังที่โดนแดดบ่อย ๆ เช่น ใบหน้า ลำคอ และมือ มะเร็งชนิดนี้ เติบโตช้าและไม่ค่อยกระจายไปยังส่วนอื่น ๆ ของร่างกาย

2. Squamous cell carcinoma เป็นมะเร็งผิวหนังที่พบมากเป็นอันดับที่ 2 มักปรากฏบนผิวหนัง บริเวณที่โดนแสงแดดเหมือนกัน แต่สามารถพัฒนาไปยังส่วนอื่น ๆ ของร่างกายได้

3. Melanoma เป็นมะเร็งผิวหนังชนิดร้ายแรงที่พบมากที่สุดอันดับ 3 โดยพัฒนาในเซลล์เมลานोไซต์ที่ทำหน้าที่ในการสร้างเม็ดสีบนผิวหนัง และส่วนที่เป็นผิวหนังปกติทางกลุ่มเราได้มีการเก็บข้อมูลด้วยตนเอง นั่นคือ

4. Normal skin - ผิวหนังที่มีลักษณะปกติ ไม่มีอาการลักษณะของโรคใดๆ

3.1.2 การเตรียมข้อมูล

ทำการแบ่งชุดข้อมูลจาก dataset จากขั้นการเตรียมข้อมูลข้างต้นออกเป็น 2 ชุดตามที่กล่าวไว้ข้างต้น คือ ชุดภาพที่ไม่มีการปรับแต่งลบขน และชุดภาพที่มีการปรับแต่งลบขน โดยรายละเอียดของ train, test และ valid มีดังนี้

1. Train เป็นข้อมูลที่ใช้สอน AI สำหรับการวินิจฉัยข้อมูล คิดเป็น 80 % ของรูปทั้งหมด
2. Test เป็นข้อมูลที่เหมือนข้อสอบสำหรับ AI เพื่อใช้ทดสอบหรือตรวจสอบผลความแม่นยำ คิดเป็น 10 % ของรูปทั้งหมด
3. Valid เป็นข้อมูลที่เหมือนเป็นแบบฝึกหัดให้กับ AI เพื่อใช้ฝึกให้ AI มีประสิทธิภาพที่ดีขึ้น คิดเป็น 10 % ของรูปทั้งหมด สรุปเป็นตารางดังนี้

Disease	Train (80%)	Test (10%)	Valid (10%)	รวม
Basal cell carcinoma	314	39	39	394
Squamous cell carcinoma	157	20	20	197
Melanoma	364	45	45	454
Normal skin	74	10	9	93
รวม	909	114	113	1136

ตารางสรุปจำนวนภาพกลุ่มชุดข้อมูล dataset ที่ไม่มีการปรับแต่งลบเส้นขน

Disease	Train (80%)	Test (10%)	Valid (10%)	รวม
Basal cell carcinoma	314	39	39	392
Squamous cell carcinoma	157	20	20	197
Melanoma	364	45	45	454
Normal skin	74	9	8	91
รวม	909	113	112	1134

ตารางสรุปจำนวนภาพกลุ่มชุดข้อมูล dataset ที่มีการปรับแต่งลบเส้นขนด้วย DHR

โดยสรุป dataset ที่กลุ่มเราใช้ทั้งหมดมี กลุ่มภาพที่ไม่ลบเส้นขน 1136 รูป และกลุ่มภาพที่ไม่ลบเส้นขน 1134 รูป เพื่อเปรียบเทียบประสิทธิภาพ โดยทำการลบเส้นขนออกจากภาพด้วย DHR จะทำการปรับภาพที่ได้ให้เป็นขาวดำก่อน (grayScale) จากนั้นจะกรองภาพเลือกเฉพาะส่วนที่เป็นเส้นขน (blackhat) และ ลบออกจากภาพดั้งเดิม (dst) และนำผลลัพธ์จากทั้งสองชุดมาเปรียบเทียบประสิทธิภาพการทำนาย

```

# DHR ลบเส้นขนออกจากกรุปใน # DHR ลบเส้นขนออกจากกรุปใน train_path

# กำหนดเส้นทางไปยังโฟลเดอร์ที่มีรูปภาพ
folder_path = train_path

# วนซ้ำทุกไฟล์ในโฟลเดอร์
for root, dirs, files in os.walk(folder_path):
    for file_name in files:
        # ตรวจสอบว่าไฟล์เป็นรูปภาพหรือไม่
        if file_name.endswith(".jpg") or file_name.endswith(".png"):

            full_file_path = os.path.join(root, file_name)

            img = cv2.imread(full_file_path)

            # Resize the image
            src = cv2.resize(img, (224, 224))

            # Convert the original image to grayscale
            grayScale = cv2.cvtColor( src, cv2.COLOR_RGB2GRAY )

            # Kernel for the morphological filtering
            kernel = cv2.getStructuringElement(1,(17,17))

            # Perform the blackHat filtering on the grayscale image to find the hair countours
            blackhat = cv2.morphologyEx(grayScale, cv2.MORPH_BLACKHAT, kernel)

            # intensify the hair countours in preparation for the inpainting algorithm
            ret,thresh2 = cv2.threshold(blackhat,10,255,cv2.THRESH_BINARY)

            # inpaint the original image depending on the mask
            dst = cv2.inpaint(src,thresh2,1,cv2.INPAINT_TELEA)

            # บันทึกรูปภาพ
            cv2.imwrite(os.path.join(root, "processed_" + file_name), dst)

```

ภาพโค้ดที่ใช้ในการลบเส้นขนจากภาพด้วย DHR

3.2 การสร้างโมเดล

3.2.1 การดาวน์โหลดและรวบรวมชุดข้อมูล (Downloading Datasets)

เป็นขั้นตอนแรกในการเตรียมข้อมูลสำหรับการสร้างโมเดล โดยต้องค้นหาและดึงชุดข้อมูลที่เกี่ยวข้องกับงานที่ต้องการทำ

3.2.2 การนำเข้าไลบรารีและชุดข้อมูล (Import Library and Dataset)

เป็นการโหลดชุดข้อมูลเข้ามาในโปรแกรมและนำเข้าไลบรารีที่จำเป็นสำหรับการประมวลผล

3.2.3 การจัดเตรียมข้อมูล (Data Preparation)

เป็นขั้นตอนการปรับแต่งและจัดรูปแบบข้อมูลให้อยู่ในรูปแบบที่เหมาะสมสำหรับการสร้างโมเดล

3.2.3.1 การกำหนดพารามิเตอร์สำหรับแบ่งชุดข้อมูล (Set Parameter to Split Data)

เป็นการกำหนดสัดส่วนการแบ่งชุดข้อมูลออกเป็นส่วนสำหรับการฝึกและทดสอบโมเดล

3.2.3.2 การตั้งค่าพารามิเตอร์พิเศษ (Set up Custom Parameter)

เพื่อปรับแต่งและกำหนดค่าพิเศษของขั้นตอนการสร้างโมเดลให้เหมาะสม

3.2.3.3 การรันฟังก์ชันสร้างชุดข้อมูลสำหรับภาพ (Run ImageDataGenerator)

เป็นขั้นตอนสร้างและจัดเตรียมชุดข้อมูลภาพสำหรับการฝึกโมเดล

3.2.3.4 การกำหนดสถาปัตยกรรมของโมเดล (Create Model)

เป็นการออกแบบและกำหนดโครงสร้างของโมเดลที่ต้องการสร้าง

3.2.3.5 การเริ่มกระบวนการฝึกโมเดล (Train Model Initiate)

เพื่อเตรียมความพร้อมก่อนเริ่มฝึกโมเดลจริง

3.2.3.6 การฝึกโมเดล (Train Model)

เป็นขั้นตอนปรับปรุงและพัฒนาประสิทธิภาพของโมเดลผ่านการใช้ชุดข้อมูลสำหรับการฝึกที่จัดเตรียมไว้

3.2.3.7 การประเมินโมเดลบนชุดข้อมูลทดสอบ (Evaluate Model on Test Set)

เพื่อวัดประสิทธิภาพของโมเดลที่สร้างขึ้น

3.2.3.8 การประเมินโมเดลอย่างละเอียด (Evaluate Model in Detail)

เป็นการวิเคราะห์ผลประสิทธิภาพด้วยสถิติและดัชนีชี้วัดอื่นๆ

3.2.3.9 การบันทึกและโหลดโมเดลที่สร้างขึ้น (Save & Load Model)

เพื่อสามารถนำไปใช้งานภายหลังได้

3.2.3.10 การทดสอบและใช้งานโมเดลกับข้อมูลจริง (Deploy and Test with Real Images)

เป็นการนำโมเดลมาประยุกต์ใช้กับข้อมูลภาพจริงเพื่อประเมินความสามารถ

3.3 คุณสมบัติของแต่ละโมเดล

3.3.1 ResNet-50 เป็นโมเดลที่มีความลึกถึง 50 layers ในสถาปัตยกรรมของโครงข่ายประสาทเทียม (CNN) โดยถูกออกแบบมาเพื่อการจำแนกประเภทวัตถุในภาพและการตรวจจับวัตถุ โมเดลนี้มีความสามารถในการเรียนรู้ลึกของข้อมูลภาพที่ซับซ้อนและถูกใช้งานอย่างแพร่หลายในหลากหลายงานที่ต้องการความแม่นยำและประสิทธิภาพสูง

- ข้อดีของ ResNet-50
 - มีความลึกสูงถึง 50 layers ทำให้มีความสามารถในการเรียนรู้ลึกของข้อมูลภาพที่ซับซ้อนได้ดี
 - ลดปัญหา Gradient Vanishing/Exploding ที่เป็นปัญหาในการฝึกโมเดลลึก
- ข้อเสีย ResNet-50
 - มีจำนวนพารามิเตอร์และการใช้หน่วยความจำที่มาก ต้องใช้ทรัพยากรคอมพิวเตอร์มากในการฝึกและใช้งาน
 - การฝึกและทดสอบอาจใช้เวลานาน

3.3.2 ASNet-Large เป็นโมเดลที่ได้รับความนิยมในการจำแนกและตรวจจับวัตถุในภาพ โดยเป็นผลลัพธ์ของกระบวนการค้นหาโครงสร้างของโมเดลอัตโนมัติโดยใช้เทคนิค Neural Architecture Search (NAS) ซึ่งมีความสามารถในการจัดการกับภาพที่มีข้อมูลซับซ้อนและหลายระดับได้อย่างมีประสิทธิภาพ

- ข้อดีของ NASNet-Large
 - มีความสามารถในการจัดการกับภาพที่มีข้อมูลซับซ้อนและหลายระดับ
 - มีความแม่นยำสูงในการจำแนกและตรวจจับวัตถุ
- ข้อเสียของ NasNet-Large

- ต้องใช้ทรัพยากรคอมพิวเตอร์มากในการฝึกและใช้งานเนื่องจากมีขนาดและความซับซ้อนของโมเดลที่ใหญ่มาก
 - การฝึกและทดสอบอาจใช้เวลานานและต้องใช้พลังงานมาก
- 3.3.3 MobileNetV3Small เป็นโมเดลที่ออกแบบมาเพื่อใช้งานบนอุปกรณ์เคลื่อนที่ โดยมีขนาดกะทัดรัดและมีประสิทธิภาพในการจำแนกวัตถุในภาพ โมเดลนี้ได้ถูกพัฒนาโดยใช้เทคนิคต่าง ๆ เช่น Inverted Residual Blocks และ SE-Blocks เพื่อเพิ่มประสิทธิภาพและลดขนาดของโมเดล
- ข้อดีของ MobileNetV3Small
 - ขนาดเล็กและมีน้ำหนักเบาบางเหมาะสำหรับการใช้งานบนอุปกรณ์เคลื่อนที่
 - มีประสิทธิภาพการจำแนกสูงและมีความเร็วในการฝึกและใช้งาน
 - ข้อเสียของ MobileNetV3Small
 - ความแม่นยำอาจจะไม่สูงเท่ากับโมเดลที่มีขนาดใหญ่กว่า
 - มีข้อจำกัดในการใช้กับงานที่ต้องการความซับซ้อนสูง
- 3.3.4 EfficientNetV2L เป็นโมเดลที่พัฒนาต่อยอดมาจาก EfficientNet โดยมีขนาดใหญ่ขึ้นและมีประสิทธิภาพสูงกว่า EfficientNet เดิม โมเดลนี้ถูกออกแบบมาเพื่อใช้กับงานที่ต้องการความแม่นยำสูงโดยใช้เทคนิคการออกแบบโครงสร้างที่เหมาะสม
- ข้อดีของ EfficientNetV2L
 - มีความแม่นยำในการจำแนกสูง เนื่องจากเป็นโมเดลขนาดใหญ่และฝึกกับ ImageNet dataset
 - มีความสามารถในการจัดการกับปัญหาการโอเวอร์ฟิตได้ดี
 - ข้อเสียของ EfficientNetV2L
 - มีขนาดใหญ่และมีจำนวนพารามิเตอร์มากกว่า จึงต้องใช้ทรัพยากรในการฝึกและใช้งานมากขึ้น
 - มีความซับซ้อนในการปรับแต่งและต้องใช้เวลาในการฝึกนานกว่า

3.4 การทำงานของโมเดล

3.4.1 ResNet-50 :

- โมเดล ResNet-50 ถูกโหลดด้วยน้ำหนักที่ฝึกมาจาก ImageNet dataset โดยกำหนดค่า `include_top=False` เช่นเดียวกัน
- ค่าพารามิเตอร์ของโมเดล `base_model.trainable` ถูกตั้งค่าให้สามารถปรับแต่งได้ในระหว่างการฝึกอบรมใหม่

3.4.2 NASNetLarge :

- โมเดล NASNetLarge ถูกโหลดด้วยน้ำหนักที่ฝึกมาจาก ImageNet dataset โดยกำหนดค่า `include_top=False`
- ค่าพารามิเตอร์ของโมเดล `base_model.trainable` ถูกตั้งค่าให้สามารถปรับแต่งได้ในระหว่างการฝึกอบรมใหม่

3.4.3 MobileNetV3Small :

- โมเดล MobileNetV3Small ถูกโหลดด้วยน้ำหนักที่ฝึกมาจาก ImageNet dataset โดยกำหนดค่า `include_top=False` เพื่อไม่ใช้ชั้นเอาต์พุตสุดท้ายที่ออกแบบมาสำหรับ 1000 class ของ ImageNet
- ค่าพารามิเตอร์ของโมเดล `base_model.trainable` ถูกตั้งค่าให้สามารถปรับแต่งได้ในระหว่างการฝึกอบรมใหม่

3.4.4 EfficientNetV2L :

- โมเดล EfficientNetV2L ถูกโหลดด้วยน้ำหนักที่ฝึกมาจาก ImageNet dataset โดยกำหนดค่า `include_top=False` เช่นเดียวกัน
- ค่าพารามิเตอร์ของโมเดล `base_model.trainable` ถูกตั้งค่าให้สามารถปรับแต่งได้ในระหว่างการฝึกอบรมใหม่

จากนั้น โมเดลทั้ง 4 ถูกนำไปประกอบเป็นโมเดลปัญญาประดิษฐ์ใหม่ โดยมีการเพิ่มชั้นพูลลิง (GlobalAveragePooling2D) และตัวเชื่อมต่อ (Dense) เพื่อให้ได้ผลลัพธ์เป็นคลาสผลลัพธ์สุดท้ายการเพิ่มชั้นพูลลิงคือ ขนาดของข้อมูลที่ออกมาจากชั้นก่อนหน้าจะถูกลดลงเป็นขนาดเดียวกันสำหรับทุกๆ ช่อง ซึ่งทำให้ง่ายต่อการประมวลผลต่อไป

ในขั้นตอนการฝึกอบรมได้มีการใช้ชุดข้อมูลการฝึก (train_generator) และชุดข้อมูลการทดสอบ (valid_generator) และมีการปรับแต่งค่าพารามิเตอร์ของโมเดลใหม่ โดยใช้ฟังก์ชัน `model_0.fit()` ซึ่งเรียกใช้ EarlyStopping callback เพื่อป้องกันปัญหาการโอเวอร์ฟิต

ในขั้นตอนสุดท้าย ได้มีการประเมินประสิทธิภาพของโมเดลที่ผ่านการฝึกอบรมแล้วบนชุดข้อมูลทดสอบ (test_generator) โดยใช้ฟังก์ชัน `model_0.evaluate_generator()`

3.5 วัดประสิทธิภาพของโมเดล

นำทั้ง 4 model มา train เพื่อดูประสิทธิภาพ จากนั้นเปลี่ยน data เป็นภาพที่ใช้ DHR ลบเส้นขนออกจากภาพแล้ว นำมา train ทั้ง 4 model อีกครั้ง เพื่อประสิทธิภาพของแต่ละ model แล้วนำผลลัพธ์ของแต่ละโมเดลทั้งก่อนและหลังลบขนมาเปรียบเทียบกับว่าโมเดลแบบใดมีประสิทธิภาพดีที่สุด ซึ่งสามารถดูได้จาก Accuracy, Precision, Recall และ F1-score โดยอธิบายแต่ละตัวแปร ดังนี้

1. Accuracy (ความแม่นยำ) คือ อัตราส่วนของข้อมูลที่ถูกต้องทั้งหมดโดยการทำนาย ซึ่งคำนวณจาก (จำนวนข้อมูลที่ถูกต้อง / จำนวนข้อมูลทั้งหมด)

2. Precision (ความแม่นยำของการตรวจสอบ) คือ อัตราส่วนของข้อมูลที่ถูกต้องที่มีการทำนายว่าเป็น positive เทียบกับทั้งหมดที่ทำนายว่าเป็น positive ซึ่งคำนวณจาก (จำนวนข้อมูล positive ที่ถูกต้อง / จำนวนข้อมูลที่ทำนายว่าเป็น positive)
3. Recall (ความแม่นยำในการจับคืน) คือ อัตราส่วนของข้อมูลที่ถูกต้องที่มีการทำนายว่าเป็น positive จริงๆ เทียบกับจำนวนข้อมูล positive ทั้งหมด ซึ่งคำนวณจาก (จำนวนข้อมูล positive ที่ถูกต้อง / จำนวนข้อมูล positive ทั้งหมด)
4. F1-score (คะแนน F1) คือ การเฉลี่ยเอาไว้สำหรับการประเมินความแม่นยำของแบบจำลอง โดยเน้นความสมดุลระหว่าง Precision และ Recall ซึ่งคำนวณจาก $((2 * (Precision * Recall)) / (Precision + Recall))$

3.6 การปรับพารามิเตอร์

ทางกลุ่มได้มีการปรับพารามิเตอร์ของโมเดลหลังจากได้โมเดลที่มีประสิทธิภาพมากที่สุด โดยอ้างอิงตามงานวิจัย ซึ่งมีการปรับพารามิเตอร์ ดังนี้

> 2.1 Set up Custom parameter Datagen

▶

ROTATION_RANGE: 0

HORIZONTAL_FLIP: ☐

VERTICAL_FLIP: ☐

ZOOM_RANGE: 0

WIDTH_SHIFT_RANGE: 0

HEIGHT_SHIFT_RANGE: 0

LOW_BRIGHTNESS: 1

HIGH_BRIGHTNESS: 1

SHEAR_RANGE: 0

FILL_MODE: nearest

BATCH_SIZE: 32

IMAGE_SIZE: 224

[Show code](#)

ก่อนปรับพารามิเตอร์

ROTATION_RANGE: 90

HORIZONTAL_FLIP: ☒

VERTICAL_FLIP: ☒

ZOOM_RANGE: 0.2

WIDTH_SHIFT_RANGE: 0

HEIGHT_SHIFT_RANGE: 0

LOW_BRIGHTNESS: 1

HIGH_BRIGHTNESS: 1

SHEAR_RANGE: 0

FILL_MODE: nearest

BATCH_SIZE: 32

IMAGE_SIZE: 224

หลังปรับพารามิเตอร์

3.7 เขียนอธิบาย source code

source code การวินิจฉัยโรคมะเร็งของกลุ่ม

1. Import Library and Dataset

```
!pip install tensorflow # Installs TensorFlow library
!pip install keras # Installs keras library

from google.colab import drive
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers , models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers.experimental import preprocessing
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications import efficientnet_v2 , NASNetLarge
import tensorflow_hub as hub
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import warnings
import os
from datetime import datetime

%matplotlib inline

drive.mount('/content/gdrive',force_remount=True)
```

2. data preparation (การเตรียม ชุดข้อมูล)

```

train_path = "/content/gdrive/MyDrive/dataset/(DHR)SkinDataset/train"
# @param{type:"string"}
test_path = "/content/gdrive/MyDrive/dataset/(DHR)SkinDataset/test" # @param{type:"string"}
valid_path = "/content/gdrive/MyDrive/dataset/(DHR)SkinDataset/valid" #
@param{type:"string"}

```

2.1 Set up Custom parameter Datagen

```

# @title 2.1 Set up Custom parameter Datagen { display-mode: "form" }
ROTATION_RANGE = 0 # @param {type:"number"}
HORIZONTAL_FLIP = False # @param{type:"boolean"}
VERTICAL_FLIP = False # @param{type:"boolean"}
ZOOM_RANGE = 0 # @param {type:"number"}
WIDTH_SHIFT_RANGE = 0 # @param {type:"number"}
HEIGHT_SHIFT_RANGE = 0 # @param {type:"number"}
LOW_BRIGHTNESS = 1 # @param {type: "number"}
HIGH_BRIGHTNESS = 1 # @param {type:"number"}
# ช่วงของความสว่างของภาพ <1 มีดลง >1 สว่างขึ้น
BRIGHTNESS_RANGE = [LOW_BRIGHTNESS, HIGH_BRIGHTNESS]
SHEAR_RANGE = 0 # @param {type: "number"}
FILL_MODE = "nearest" # @param ["constant", "nearest", "reflect"]
# constant เติมสีดำ, nearest เติมสีข้างเคียง, reflect เติมสีจากภาพที่มีการกลับด้าน

BATCH_SIZE = 32 # @param{type:"integer"}
IMAGE_SIZE = 224 # @param{type:"integer"}

```

2.2 Run ImageDataGenerator

```

# @title 2.2 Run ImageDataGenerator

train_datagen = ImageDataGenerator( width_shift_range = WIDTH_SHIFT_RANGE ,
                                     height_shift_range = HEIGHT_SHIFT_RANGE,

```

```

        rotation_range = ROTATION_RANGE ,
        horizontal_flip = HORIZONTAL_FLIP ,
        vertical_flip = VERTICAL_FLIP ,
        zoom_range = ZOOM_RANGE ,
        brightness_range = BRIGHTNESS_RANGE ,
        shear_range = SHEAR_RANGE ,
        fill_mode = FILL_MODE
    )

train_generator = train_datagen.flow_from_directory(
    directory=train_path,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical')

test_datagen = ImageDataGenerator( width_shift_range = WIDTH_SHIFT_RANGE ,
    height_shift_range = HEIGHT_SHIFT_RANGE,
    rotation_range = ROTATION_RANGE ,
    horizontal_flip = HORIZONTAL_FLIP ,
    vertical_flip = VERTICAL_FLIP ,
    zoom_range = ZOOM_RANGE ,
    brightness_range = BRIGHTNESS_RANGE ,
    shear_range = SHEAR_RANGE ,
    fill_mode = FILL_MODE
)

test_generator = test_datagen.flow_from_directory(
    directory=test_path,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    shuffle = False,
    class_mode='categorical')

```

```

valid_datagen = ImageDataGenerator( width_shift_range = WIDTH_SHIFT_RANGE ,
                                   height_shift_range = HEIGHT_SHIFT_RANGE,
                                   rotation_range = ROTATION_RANGE ,
                                   horizontal_flip = HORIZONTAL_FLIP ,
                                   vertical_flip = VERTICAL_FLIP ,
                                   zoom_range = ZOOM_RANGE ,
                                   brightness_range = BRIGHTNESS_RANGE ,
                                   shear_range = SHEAR_RANGE ,
                                   fill_mode = FILL_MODE
                                   )

valid_generator = valid_datagen.flow_from_directory(
    directory=valid_path,
    target_size=(IMAGE_SIZE, IMAGE_SIZE),
    batch_size=BATCH_SIZE,
    shuffle = False,
    class_mode='categorical')

class_names = train_generator.class_indices
class_names

CLASS_SIZE = len(class_names)

```

3. Create Model

```

import random
SEED = 42
os.environ['PYTHONHASHSEED']=str(SEED)
os.environ['TF_CUDNN_DETERMINISTIC'] = '1' # TF 2.1
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)

```

#set seed

@title เลือก Model ที่นำมาใช้ Train

```

model_name = "RestNet-50" # @param ["MobileNetV3Small", "RestNet-50", "EfficientNetV2L",
"NASNetLarge"]
if model_name == "MobileNetV3Small" :
    base_model = keras.applications.MobileNetV3Small(input_shape=(IMAGE_SIZE, IMAGE_SIZE,
3),include_top=False, weights='imagenet') # Load the MobileNetV3Small model
    base_model.trainable = True # Enable to train Pre-Trained Model
if model_name == "RestNet-50":
    base_model = ResNet50(
        input_shape = [IMAGE_SIZE , IMAGE_SIZE ] + [3], # Making the image into 3 Channel, so
concatating 3.
        weights = 'imagenet', # Default weights.
        include_top = False #
    )
    base_model.trainable = True # Enable to train Pre-Trained Model
if model_name == "EfficientNetV2L":
    base_model = efficientnet_v2.EfficientNetV2B0(include_top=False,
                                                    input_shape= [IMAGE_SIZE , IMAGE_SIZE ] + [3],
                                                    weights="imagenet")

if model_name == "NASNetLarge":
    base_model = NASNetLarge(

        include_top=False,
        input_shape = [IMAGE_SIZE , IMAGE_SIZE ] + [3],
        weights="imagenet",
    )

inputs = tf.keras.layers.Input(shape=(224, 224, 3), name="input_layer") # Define input layer

x = inputs # Apply Augmentation to the input images. this use for decrease overfit
x = base_model(x, training=False) # Pass Augmentation Data to base_model
x = layers.GlobalAveragePooling2D(name="GloPool")(x) # Create Global Average Pooling and
input x to get output tensor that value = Dense's filter. this use for decrease feature map

```



```

outputs = tf.keras.layers.Dense(CLASS_SIZE, activation="softmax", name="output_layer")(x) #
output layer that (output unit = CLASS_SIZE as image class, activation function is softmax )
model_0 = tf.keras.Model(inputs, outputs) # Create Model name model_0 that have input
layer, base model, output layer
model_0.compile(loss='categorical_crossentropy',
                 optimizer=tf.keras.optimizers.Adam(0.0001),
                 metrics=["accuracy"]) #Compile model_0 with categorical_crossentropy, adam's
learning rate = 0.0001, metric by accuracy
model_0.summary()
custom_early_stopping = EarlyStopping( # EarlyStopping is keras's callback function that stop
training model before overfitting
    monitor='val_loss', # monitor at validation loss
    patience=10,
    min_delta=0.000000001, # if validation loss is not decrease at least 0.001 in 10 time
    mode='min' # need validation loss to decrease
)

```

3. Train Model

```

start = datetime.now()

history = model_0.fit(train_generator, # Use from train generator
                     epochs=30, # training round
                     workers=0,
                     steps_per_epoch=len(train_generator), # Use from batch size that can update each
epoch for training
                     validation_data=valid_generator, # Use from test generator
                     validation_steps=len(valid_generator),
                     callbacks=[custom_early_stopping]) # Stop training when overfitting

print ('Execution Time: ',datetime.now()-start)

```

3.2. Evaluate the model on the test set

```
#@title 3.2. Evaluate the model on the test set
test_loss, test_acc = model_0.evaluate_generator(test_generator, steps=len(test_generator))
print('Test loss:', test_loss)
print('Test accuracy:', test_acc)
```

4.Evaluate Model (สรุปผล Model)

```
# Evaluate the model on the test set
start = datetime.now()
test_loss, test_acc = model_0.evaluate_generator(test_generator, steps=len(test_generator))
print('Test loss:', test_loss)
print('Test accuracy:', test_acc)
print('Test Time:',datetime.now()-start)
```

```
import sklearn as scikit_learn
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
import numpy as np
from sklearn.metrics import precision_recall_fscore_support
```

```
test_true=test_generator.classes[test_generator.index_array]
test_pred_raw = model_0.predict(test_generator)
test_pred = np.argmax(test_pred_raw, axis=1)
```

```
cm = confusion_matrix(test_true, test_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
fig, ax = plt.subplots(figsize=(10,10))
disp.plot(ax=ax,cmap=plt.cm.Blues)
plt.show()
```

```
res = []
for l in range(CLASS_SIZE):
```

```

    pres,recall,_,_ =
precision_recall_fscore_support(np.array(test_true)==l,np.array(test_pred)==l,pos_label=True,
average=None)
    res.append([l,recall[0],recall[1]])
pd.DataFrame(res,columns = ['class','specificity','sensitivity'])

from sklearn.metrics import classification_report
report = classification_report(test_true, test_pred, target_names=class_names)
print(report)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend(loc=0)
plt.figure(figsize=(6,6))

plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend(loc=0)
plt.figure(figsize=(6,6))
plt.show()

5. Save & Load h5 Model
# Save the trained model to a .h5 file

```

```

model_0.save('/content/gdrive/MyDrive/dataset/model/trainedmodel.h5')
from keras.models import load_model
model = load_model('/content/gdrive/MyDrive/dataset/model/trainedmodel.h5')
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_LATENCY] tflite_model =
converter.convert()
open("/content/gdrive/MyDrive/dataset/model/tflitemodel.tflite","wb").write
(tflite_model)
6.Deploy and Test with real image
from google.colab import files
uploaded = files.upload()
for fn in uploaded.keys():
print('User uploaded file "{name}" with length {length} bytes'.format(name=fn,
length=len(uploaded[fn])))
file_name=fn

img = image.load_img(file_name, target_size=(224, 224))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = tf.keras.applications.mobilenet_v3.preprocess_input(img_array)
# Make prediction
prediction = model.predict(img_array)
class_names = train_generator.class_indices
class_names = {v: k for k, v in class_names.items()}#reverse the class_indices dictionary
predicted_class = np.argmax(prediction, axis=1)
predicted_class_name = class_names[predicted_class[0]]
predicted_prob = np.max(prediction, axis=1)

# Print the result
print("Image name:", file_name)
print("Predicted class: {} {:.2f}%".format(predicted_class_name, predicted_prob[0]*100))
prediction_list = prediction.tolist() # Convert the NumPy array to a Python list

```

```

prediction_formatted = ["{:.2%}".format(p) for p in prediction_list[0]]
print("Prediction: ", prediction_formatted)
plt.imshow(img)
plt.title("Predicted class: {} ({:.2f}%)".format(predicted_class_name, predicted_prob[0]*100))
plt.show()
print("\n")

```

1. Import Library and Dataset

```
!pip install tensorflow # ติดตั้งไลบรารี TensorFlow
```

```
!pip install keras # ติดตั้งไลบรารี Keras
```

การนำเข้าโมดูลที่จำเป็น:

```

from google.colab import drive # นำเข้าโมดูลสำหรับโต้ตอบกับ Google Colab
import tensorflow as tf # นำเข้าไลบรารี TensorFlow และตั้งชื่อเล่นว่า tf
from tensorflow import keras # นำเข้าไลบรารี Keras จาก TensorFlow
from tensorflow.keras import layers, models
# นำเข้าโมดูลย่อยสำหรับสร้างเลเยอร์และโมเดลจาก Keras
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# นำเข้าคลาสสำหรับสร้างชุดข้อมูลภาพพร้อม augmentation แบบเรียลไทม์
from tensorflow.keras.layers.experimental import preprocessing
# นำเข้าเลเยอร์ preprocessing ทดลองสำหรับงาน เช่น การ normalization และ rescaling
from tensorflow.keras.callbacks import EarlyStopping
# นำเข้าฟังก์ชัน callback เพื่อหยุดการฝึกเมื่อประสิทธิภาพหยุดการปรับปรุง
from tensorflow.keras.preprocessing import image
# นำเข้ายูทิลิตี้สำหรับการโหลดและ preprocessing ภาพ
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
# นำเข้าโมเดล ResNet50 ที่ผ่านการฝึกอบรมมาแล้วและฟังก์ชัน preprocessing
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
# นำเข้าเลเยอร์สำหรับสร้างสถาปัตยกรรมเครือข่ายประสาทเทียม

```

```

from tensorflow.keras.models import Model

# นำเข้าคลาสสำหรับสร้างและรวมโมเดล Keras

from tensorflow.keras.applications import efficientnet_v2, NASNetLarge

# นำเข้าโมเดลที่ผ่านการฝึกอบรมมาแล้วอื่นๆ สำหรับการใช้งาน

import tensorflow_hub as hub

# นำเข้าไลบรารี TensorFlow Hub สำหรับการโหลดโมเดลที่ผ่านการฝึกอบรมมาแล้ว

from sklearn.metrics import classification_report

# นำเข้าฟังก์ชันสำหรับสร้างรายงานประเมินการจำแนกประเภทอย่างละเอียด

import matplotlib.pyplot as plt

# นำเข้าไลบรารี Matplotlib สำหรับการพล็อตและ visualize ข้อมูล

import numpy as np

# นำเข้าไลบรารี NumPy สำหรับการคำนวณเชิงตัวเลข

import pandas as pd

# นำเข้าไลบรารี Pandas สำหรับการจัดการข้อมูลและการวิเคราะห์

import warnings

# นำเข้าโมดูล warnings สำหรับจัดการ warnings ที่อาจเกิดขึ้น

import os

# นำเข้าโมดูลระบบปฏิบัติการสำหรับการโต้ตอบกับระบบไฟล์

from datetime import datetime

# นำเข้าโมดูล datetime สำหรับการทำงานกับวันที่และเวลา

```

การกำหนดค่า Matplotlib:

```

%matplotlib inline

# กำหนดค่า Matplotlib ให้แสดงพล็อตแบบ inline ภายในสภาพแวดล้อม Colab

```

การเชื่อมต่อ Google Drive:

```

drive.mount('/content/gdrive', force_remount=True)

# เชื่อมต่อ Google Drive กับไดเรกทอรี '/content/gdrive' ในสภาพแวดล้อม Colab ช่วยให้เข้าถึงไฟล์ที่เก็บไว้บน Drive

```

2. Data preparation

กำหนดเส้นทางข้อมูล:

```
train_path = "/content/gdrive/MyDrive/dataset/(DHR)SkinDataset/train"
```

```
# เส้นทางไปยังโฟลเดอร์ข้อมูลฝึกอบรม (train) บน Google Drive
```

```
test_path = "/content/gdrive/MyDrive/dataset/(DHR)SkinDataset/test"
```

```
# เส้นทางไปยังโฟลเดอร์ข้อมูลทดสอบ (test) บน Google Drive
```

```
valid_path = "/content/gdrive/MyDrive/dataset/(DHR)SkinDataset/valid"
```

```
# เส้นทางไปยังโฟลเดอร์ข้อมูลตรวจสอบ (validation) บน Google Drive
```

2.1 Set up Custom parameter Datagen

กำหนดค่าพารามิเตอร์สำหรับ ImageDataGenerator ซึ่งใช้สำหรับ augmentation ข้อมูลภาพ:

```
ROTATION_RANGE = 0 # ช่วงการหมุนภาพ (องศา)
```

```
HORIZONTAL_FLIP = False # ระบุว่าจะพลิกภาพแนวนอนหรือไม่
```

```
VERTICAL_FLIP = False # ระบุว่าจะพลิกภาพแนวตั้งหรือไม่
```

```
ZOOM_RANGE = 0 # ช่วงการซูมภาพ
```

```
WIDTH_SHIFT_RANGE = 0 # ช่วงการเลื่อนภาพตามแนวกว้าง
```

```
HEIGHT_SHIFT_RANGE = 0 # ช่วงการเลื่อนภาพตามแนวสูง
```

```
LOW_BRIGHTNESS = 1 # ความสว่างต่ำสุด (ค่าต่ำกว่า 1 ทำให้ภาพมืดลง)
```

```
HIGH_BRIGHTNESS = 1 # ความสว่างสูงสุด (ค่ามากกว่า 1 ทำให้ภาพสว่างขึ้น)
```

```
BRIGHTNESS_RANGE = [LOW_BRIGHTNESS, HIGH_BRIGHTNESS] # ช่วงความสว่างของภาพ
```

```
SHEAR_RANGE = 0 # ช่วงการเอียงภาพ
```

```
FILL_MODE = "nearest" # วิธีการเติมพื้นที่ที่ขาดหาย ["constant", "nearest", "reflect"]
```

```
BATCH_SIZE = 32 # ขนาดของกลุ่มข้อมูลที่จะใช้ในการฝึก
```

IMAGE_SIZE = 224 # ขนาดของภาพที่ใช้ในโมเดล

2.2 Run ImageDataGenerator

โค้ดส่วนนี้ใช้สร้างและรัน ImageDataGenerator เพื่อเตรียมข้อมูลสำหรับการฝึกอบรม (train) การทดสอบ (test) และการตรวจสอบ (validation) ของโมเดล

สร้าง ImageDataGenerator:

- 'train_datagen': สร้าง ImageDataGenerator สำหรับข้อมูลฝึกอบรมโดยใช้พารามิเตอร์ที่กำหนดไว้ในส่วน 2.1
- 'test_datagen': สร้าง ImageDataGenerator สำหรับข้อมูลทดสอบโดยใช้พารามิเตอร์เดียวกันกับข้อมูลฝึกอบรม
- 'valid_datagen': สร้าง ImageDataGenerator สำหรับข้อมูลตรวจสอบโดยใช้พารามิเตอร์เดียวกันกับข้อมูลฝึกอบรม

สร้าง ImageData Flow:

- 'train_generator': สร้างตัวแปร train_generator โดยใช้ฟังก์ชัน 'flow_from_directory' ของ 'train_datagen'
- 'directory': กำหนดเส้นทางไปยังโฟลเดอร์ข้อมูลฝึกอบรม (train_path)
- 'target_size': กำหนดขนาดภาพที่ต้องการ (IMAGE_SIZE x IMAGE_SIZE)
- 'batch_size': กำหนดจำนวนภาพในแต่ละ batch (BATCH_SIZE)
- 'class_mode': กำหนดว่าข้อมูลมีหลายคลาส (categorical)
- 'test_generator': สร้างตัวแปร test_generator ในลักษณะเดียวกับ train_generator แต่โหลดข้อมูลจากโฟลเดอร์ทดสอบ (test_path) และไม่สุ่มลำดับข้อมูล
- 'valid_generator': สร้างตัวแปร valid_generator ในลักษณะเดียวกับ train_generator แต่โหลดข้อมูลจากโฟลเดอร์ตรวจสอบ (valid_path) และไม่สุ่มลำดับข้อมูล

ดึงข้อมูลคลาส:

- 'class_names': ดึงชื่อคลาสจาก train_generator

- 'CLASS_SIZE': กำหนดตัวแปร CLASS_SIZE โดยนับจำนวนคลาสจาก 'class_names'

3. Create Model

import random: # โหลดโมดูล random สำหรับการสุ่มตัวเลข

SEED = 42: # กำหนดค่า seed สุ่มเป็น 42

os.environ['PYTHONHASHSEED'] = str(SEED): # ตั้งค่า seed สำหรับ Python hash

os.environ['TF_CUDNN_DETERMINISTIC'] = '1':

ตั้งค่า seed สำหรับ TensorFlow CUDNN (เฉพาะ TensorFlow 2.1)

random.seed(SEED): # กำหนดค่า seed สุ่มสำหรับ Python

np.random.seed(SEED): # กำหนดค่า seed สุ่มสำหรับ NumPy

tf.random.set_seed(SEED): # กำหนดค่า seed สุ่มสำหรับ TensorFlow

เลือก Model ที่นำมาใช้ Train

โค้ด Python นี้ใช้สร้างโมเดลสำหรับการจำแนกประเภทรูปภาพโดยใช้โมเดล pre-trained และปรับแต่งโมเดล (Fine-Tuning) กับชุดข้อมูลเฉพาะของคุณ

เลือกโมเดล (Choose Model) โดยโค้ดใช้ตัวเลือกแบบดรอปดาวน์ ที่ต้องการ 4 แบบ โมเดลที่เลือกจะถูกเก็บไว้ในตัวแปร model_name

- MobileNetV3Small
- ResNet-50
- EfficientNetV2L
- NASNetLarge

ชื่อโมเดล pre-trained (Load Pre-Trained Model)

โค้ดใช้ if-elif statements เพื่อโหลดโมเดลที่เลือกใน model_name โมเดลจะถูกโหลดด้วย arguments ดังนี้:

- input_shape: กำหนดขนาดของข้อมูลรูปภาพอินพุต (ความสูง, ความกว้าง, ช่องสี)
- include_top=False: ข้ามขั้นตอนการจับหมวดหมู่สุดท้ายของโมเดล pre-trained เพื่อให้คุณสามารถเพิ่มขั้นตอนการจับหมวดหมู่แบบกำหนดเองสำหรับงานเฉพาะของคุณ
- weights='imagenet': โหลดน้ำหนัก pre-trained ของโมเดลที่ฝึกกับชุดข้อมูล ImageNet

โมเดลที่โหลดแล้วจะถูกเก็บไว้ในตัวแปร base_model

`base_model.trainable = True:`

อนุญาตให้ปรับแต่งน้ำหนักของชั้นในโมเดล pre-trained

สร้างโมเดล (Create Model):

- โค้ดสร้างเลเยอร์อินพุต (Input Layer) โดยกำหนดขนาดของข้อมูลรูปภาพอินพุต
- ส่งข้อมูลอินพุตผ่านโมเดล pre-trained โดยตั้งค่า `training=False` เนื่องจากเราไม่ได้ต้องการฝึกโมเดล pre-trained
- สร้างเลเยอร์ Global Average Pooling เพื่อลดขนาดของ feature map
- สร้างเลเยอร์ Dense สำหรับการจัดหมวดหมู่ โดยกำหนดจำนวน output units เท่ากับจำนวนคลาสของรูปภาพ และใช้ฟังก์ชันการกระตุ้น softmax
- สร้างโมเดลโดยใช้เลเยอร์อินพุต โมเดล pre-trained และเลเยอร์ Dense

คอมไพล์โมเดล (Compile Model):

- โค้ดกำหนดฟังก์ชันการสูญเสีย (loss function) ตัวปรับแต่ง (optimizer) และเมตริก (metric) สำหรับการฝึกโมเดล

สรุปโมเดล (Print Model Summary):

- โค้ดแสดงรายละเอียดของโมเดล เช่น จำนวนเลเยอร์ จำนวนพารามิเตอร์

กำหนดการหยุดการฝึก (Early Stopping):

- โค้ดใช้ `EarlyStopping` callback เพื่อหยุดการฝึกโมเดลก่อนเกิด Overfitting

โค้ด Python นี้แสดงตัวอย่างการใช้โมเดล pre-trained และการปรับแต่งโมเดล (Fine-Tuning) สำหรับการจำแนกประเภทรูปภาพ

3.Train Model

โค้ด Python นี้ใช้ฝึกโมเดลสำหรับการจำแนกประเภทรูปภาพ โดยใช้โมเดล pre-trained และปรับแต่งโมเดล (Fine-Tuning) กับชุดข้อมูลเฉพาะของคุณ

ฝึกโมเดล:

โค้ดใช้ฟังก์ชัน `model_0.fit()` ฝึกโมเดล โดยระบุ:

- `train_generator`: generator สำหรับข้อมูลฝึก
- `epochs=30`: จำนวนรอบการฝึก (30 รอบ)
- `workers=0`: จำนวน worker processes (0 = ใช้ CPU หลัก)

- `steps_per_epoch=len(train_generator)`: จำนวน steps ในแต่ละ epoch
- `validation_data=valid_generator`: generator สำหรับข้อมูลทดสอบ
- `validation_steps=len(valid_generator)`: จำนวน steps ของข้อมูลทดสอบ
- `callbacks=[custom_early_stopping]`: callback function

แสดงระยะเวลา:

- โค้ดคำนวณระยะเวลาฝึกโมเดล โดยใช้ `datetime.now()` ลบ start แสดงผลลัพธ์
"Execution Time: " # โค้ดนี้ฝึกโมเดลโดยใช้ generator กำหนดจำนวนรอบ ระยะเวลา
และ callback function

3.2. Evaluate the model on the test set

โค้ด Python นี้ใช้ประเมินประสิทธิภาพโมเดลบนชุดข้อมูลทดสอบ

ประเมินโมเดล:

โค้ดใช้ฟังก์ชัน `model_0.evaluate_generator()`:

ประเมินประสิทธิภาพโมเดลบนชุดข้อมูลทดสอบ(`test_generator`)

`steps=len(test_generator)`:

ระบุจำนวน steps ที่จะประเมิน

แสดงผลลัพธ์:

- Test loss คือ ค่า loss บนชุดข้อมูลทดสอบ
- Test accuracy คือ ค่า accuracy บนชุดข้อมูลทดสอบ

โค้ดนี้ประเมินประสิทธิภาพโมเดลบนชุดข้อมูลทดสอบและแสดงผลลัพธ์ "Test loss" และ "Test accuracy"

4. Evaluate Model (สรุปผล Model) :

โค้ด Python นี้ใช้สำหรับสรุปผลการประเมินโมเดล แบ่งเป็น 5 ส่วน

ประเมินประสิทธิภาพเบื้องต้น:

- ประเมินโมเดลบนชุดข้อมูลทดสอบ (`test_generator`)
- แสดงผลลัพธ์ "Test loss" และ "Test accuracy"
- จับเวลาการประเมิน

สร้าง Confusion Matrix:

- ใช้ไลบรารี sklearn สร้าง Confusion Matrix
- แสดงผลลัพธ์ Confusion Matrix ในรูปแบบกราฟ

คำนวณค่าประเมินเพิ่มเติม:

- คำนวณค่า Precision, Recall, F1-score, Specificity, Sensitivity
- แสดงผลลัพธ์ในตารางข้อมูล (DataFrame)

รายงานประสิทธิภาพโมเดล:

- ใช้ฟังก์ชัน classification_report สร้างรายงาน
- แสดงค่า Precision, Recall, F1-score, Support

พล็อตประวัติการฝึกโมเดล:

- พล็อตกราฟแสดง Accuracy และ Loss โดยแยก Training และ Validation

4.1. ประเมินประสิทธิภาพเบื้องต้น:

model_0.evaluate_generator: # ฟังก์ชันสำหรับประเมินโมเดล

test_generator: # ข้อมูลทดสอบ

steps=len(test_generator): # ระบุจำนวน steps ที่จะประเมิน

print: # แสดงผลลัพธ์ "Test loss", "Test accuracy", และระยะเวลาการประเมิน

4.2. สร้าง Confusion Matrix :

scikit_learn : # ไลบรารีสำหรับ Machine Learning

confusion_matrix : # ฟังก์ชันสำหรับสร้าง Confusion Matrix

ConfusionMatrixDisplay : # แสดงผลลัพธ์ Confusion Matrix

test_true : # คลาสที่แท้จริง

test_pred_raw : # ผลลัพธ์ดิบจากโมเดล

test_pred : # คลาสที่โมเดลทำนาย

plt.show : # แสดงกราฟ

4.3. คำนวณค่าประเมินเพิ่มเติม :

precision_recall_fscore_support : คำนวณค่า Precision, Recall, F1-score

specificity : 1 - recall คลาสอื่น

sensitivity : recall

pd.DataFrame

5. Save & Load h5 Model

โค้ด Python นี้ใช้สำหรับบันทึกโมเดลที่ฝึกเสร็จแล้ว โหลดโมเดลกลับมา แปลงโมเดลเป็นรูปแบบ TensorFlow Lite (TFLite) และบันทึกโมเดล TFLite

บันทึกโมเดล:

```
model_0.save('/content/gdrive/MyDrive/dataset/model/trainedmodel.h5'):
```

บันทึกโมเดล model_0 เป็นไฟล์ trainedmodel.h5 ในโฟลเดอร์

โหลดโมเดล:

```
from keras.models import load_model:
```

นำเข้าฟังก์ชัน load_model จากไลบรารี Keras

```
model = load_model('/content/gdrive/MyDrive/dataset/model/trainedmodel.h5'):
```

โหลดโมเดล trainedmodel.h5 เก็บไว้ในตัวแปร model

แปลงโมเดลเป็น TFLite:

```
converter = tf.lite.TFLiteConverter.from_keras_model(model):
```

สร้าง TFLiteConverter จากโมเดล model

```
converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_LATENCY]:
```

ตั้งค่าการเพิ่มประสิทธิภาพ โมเดล เน้นการลด latency (ความหน่วง)

```
tf_lite_model = converter.convert():
```

แปลงโมเดลเป็นรูปแบบ TFLite เก็บไว้ในตัวแปร tf_lite_model

บันทึกโมเดล TFLite:

```
open("/content/gdrive/MyDrive/dataset/model/tflitemodel.tflite","wb").write(tf_lite_model)
```

บันทึกโมเดล tf_lite_model เป็นไฟล์ tflitemodel.tflite ในโฟลเดอร์ /content/gdrive/MyDrive

6.deploy and test with real image

การนำโมเดลและการแก้ไขที่พัฒนาขึ้นมาใช้งานจริงกับภาพจริง เพื่อทดสอบและตรวจสอบประสิทธิภาพ

```
from google.colab import files:
```

เรียกใช้งานโมดูล files จากไลบรารี google.colab เพื่อใช้ในการอัปโหลดไฟล์

```

uploaded = files.upload():
# เรียกใช้งานฟังก์ชัน upload() จากโมดูล files เพื่ออัปโหลดไฟล์จากเครื่อง
for fn in uploaded.keys():
# วนลูปผ่านไฟล์ที่อัปโหลดขึ้นมาในตัวแปร uploaded
print('User uploaded file "{name}" with length {length} bytes'.format(name=fn,
length=len(uploaded[fn]])):
# แสดงข้อความบอกว่าไฟล์ได้ถูกอัปโหลดขึ้นมา และแสดงชื่อของไฟล์พร้อมกับขนาดของไฟล์นั้นๆ
ในหน่วยไบต์
file_name=fn:
# เก็บชื่อของไฟล์ที่อัปโหลดขึ้นมาลงในตัวแปร file_name สำหรับการใช้งานต่อไป

```

โค้ดที่ให้มีการทำงานเกี่ยวกับการทำนายคลาสของภาพโดยใช้โมเดล

```

img = image.load_img(file_name, target_size=(224, 224)):
# โหลดภาพจากไฟล์ที่ระบุ (ที่อัปโหลดมาก่อนหน้านี้) และปรับขนาดให้เป็นขนาด 224x224
เนื่องจากโมเดล MobileNetV3 ที่ใช้ต้องการภาพขนาดเท่านี้
img_array = image.img_to_array(img):
# แปลงภาพเป็น NumPy array เพื่อให้ง่ายต่อการประมวลผล
img_array = np.expand_dims(img_array, axis=0):
# เพิ่มมิติให้กับภาพออกไปในมิติที่ 0 เพื่อให้เหมาะกับการใช้งานกับโมเดลที่รับ input ในรูปแบบ
batch
img_array = tf.keras.applications.mobilenet_v3.preprocess_input(img_array):
# ทำการประมวลผลภาพเพื่อให้เข้ากับการตรวจจับและคลาสของโมเดล MobileNetV3
prediction = model.predict(img_array):
# ทำนายคลาสของภาพโดยใช้โมเดล MobileNetV3 ที่ถูกฝึกอบอุ้นไว้แล้ว
class_names = train_generator.class_indices:
# นำเอาชื่อคลาสจาก train_generator ที่ได้จากการสร้างข้อมูลสำหรับการฝึกโมเดล
predicted_class = np.argmax(prediction, axis=1):
# หากคลาสที่ถูกทำนายโดยมีความน่าจะเป็นสูงสุด
predicted_class_name = class_names[predicted_class[0]]:
# แปลงเลขคลาสที่ทำนายได้ให้อยู่ในรูปของชื่อคลาส
predicted_prob = np.max(prediction, axis=1):
# หากความน่าจะเป็นสูงสุดที่ถูกทำนาย
print("Image name:", file_name):
# แสดงชื่อของไฟล์ที่ใช้ทำนาย

```

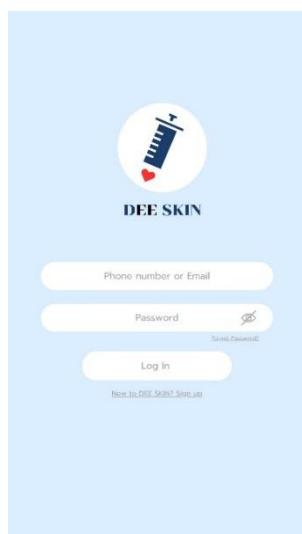
```

print("Predicted class: {} {:.2f}%".format(predicted_class_name,
predicted_prob[0]*100)):
# แสดงคลาสที่ทำนายได้พร้อมกับความน่าจะเป็น
prediction_list = prediction.tolist():
# แปลงผลการทำนายจาก NumPy array เป็นรายการ Python
prediction_formatted = ["{:.2f}%".format(p) for p in prediction_list[0]]:
# จัดรูปแบบของผลการทำนายให้อยู่ในรูปแบบเปอร์เซ็นต์
print("Prediction: ", prediction_formatted):
# แสดงผลการทำนายในรูปแบบของเปอร์เซ็นต์
plt.imshow(img):
# แสดงภาพที่ใช้ทำนาย
plt.title("Predicted class: {} {:.2f}%".format(predicted_class_name,
predicted_prob[0]*100)):
# กำหนดชื่อเรื่องของกราฟที่แสดงผลการทำนาย
plt.show():
# แสดงกราฟที่มีภาพและชื่อเรื่องที่กำหนด

```

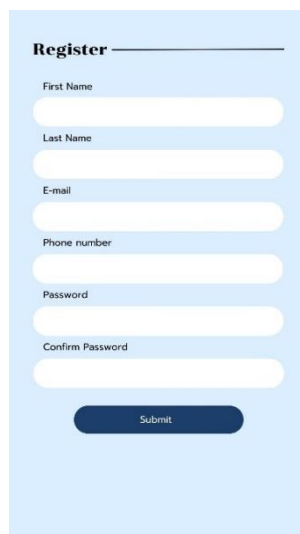
ดังนั้น โค้ดนี้ทำการอ่านภาพจากไฟล์ที่อัปโหลดขึ้นมาก่อนหน้านี้ แล้วทำการทำนายคลาสของภาพด้วยโมเดล และแสดงผลการทำนายในรูปแบบของชื่อคลาสและความน่าจะเป็นของการทำนายนั้นๆ พร้อมกับแสดงภาพที่ใช้ทำนายไปยังผู้ใช้ใน Colab โดยใช้ matplotlib.pyplot

3.8 การออกแบบ UX/UI ของ Application



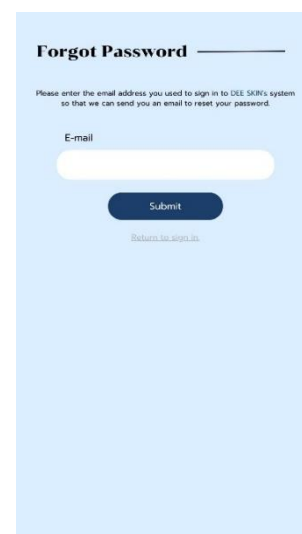
The Log In screen features a light blue background. At the top center is the DEE SKIN logo, which consists of a circular icon with a blue and white design and the text 'DEE SKIN' below it. Below the logo are two input fields: 'Phone number or Email' and 'Password'. The Password field has a small eye icon to its right. Below these fields is a 'Log In' button. At the bottom, there is a link that says 'New to DEE SKIN? Sign up'.

หน้า Log In



The Register screen has a light blue background. At the top, it says 'Register' followed by a horizontal line. Below this are five input fields: 'First Name', 'Last Name', 'E-mail', 'Phone number', and 'Password'. Below the Password field is a 'Confirm Password' field. At the bottom center is a 'Submit' button.

หน้า Register



The Forgot Password screen has a light blue background. At the top, it says 'Forgot Password' followed by a horizontal line. Below this is a paragraph of text: 'Please enter the email address you used to sign in to DEE SKIN's system so that we can send you an email to reset your password.' Below the text is an 'E-mail' input field. Below the input field is a 'Submit' button. At the bottom, there is a link that says 'Return to Sign In'.

หน้า Forget password

หน้า Log In

เป็นกระบวนการที่ผู้ใช้งานต้องดำเนินการเพื่อเข้าสู่ระบบในแอปพลิเคชัน โดยประกอบด้วยฟังก์ชัน ดังนี้

- ช่องกรอก Phone Number or Email : ผู้ใช้ต้องกรอกข้อมูลเบอร์โทรศัพท์หรืออีเมลของตนเองเพื่อทำการเข้าสู่ระบบ
- ช่องกรอก Password : ผู้ใช้ต้องกรอกรหัสผ่านที่ถูกต้องเพื่อทำการเข้าสู่ระบบ
- Forgot Password : ผู้ใช้สามารถคลิกที่ปุ่มนี้เพื่อเข้าสู่กระบวนการรีเซตรหัสผ่าน
- ปุ่ม Log In : ผู้ใช้สามารถคลิกที่ปุ่มนี้เพื่อทำการเข้าสู่ระบบในแอปพลิเคชัน
- Sign up : ผู้ใช้สามารถคลิกที่ปุ่มนี้เพื่อเข้าสู่หน้าที่ใช้ในการสมัครสมาชิกในแอปพลิเคชัน

หน้า Register

เป็นหน้าที่ใช้สำหรับลงทะเบียนเพื่อสร้างบัญชีใหม่ในแอปพลิเคชัน โดยประกอบด้วยฟังก์ชัน ดังนี้

- First Name : ให้ผู้ใช้ใส่ชื่อจริงของตนเองเพื่อใช้ในการลงทะเบียน
- Last Name : ให้ผู้ใช้ใส่นามสกุลของตนเองเพื่อใช้ในการลงทะเบียน
- E-mail : ให้ผู้ใช้ใส่ที่อยู่อีเมลของตนเองเพื่อใช้ในการติดต่อและลงทะเบียน
- Phone Number : ให้ผู้ใช้ใส่หมายเลขโทรศัพท์ของตนเองเพื่อใช้ในการติดต่อและลงทะเบียน
- Password : ให้ผู้ใช้ตั้งรหัสผ่านสำหรับบัญชีของตน
- Confirm Password : ให้ผู้ใช้ใส่รหัสผ่านอีกครั้งเพื่อยืนยันความถูกต้องของรหัสผ่านที่ตั้งไว้
- Submit : ให้ผู้ใช้คลิกที่ปุ่มนี้เพื่อส่งข้อมูลลงทะเบียนและสร้างบัญชีใหม่ในระบบ

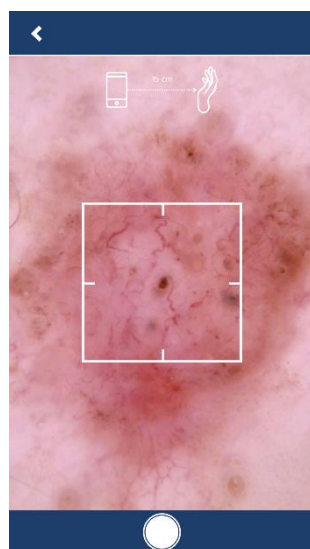
หลังจากคลิกที่ปุ่ม Submit แล้ว ระบบจะทำการตรวจสอบข้อมูลที่ใช้กรอกเพื่อดำเนินการลงทะเบียน โดยในกรณีที่ข้อมูลถูกต้องและครบถ้วน บัญชีผู้ใช้จะถูกสร้างขึ้นในระบบโดยอัตโนมัติและผู้ใช้สามารถเข้าสู่ระบบด้วยบัญชีที่ลงทะเบียนได้ทันที

หน้า Forgot Password

เมื่อผู้ใช้กดไปที่หน้านี้ จะสามารถกรอก Email เพื่อใช้ในการยืนยันตัวตนได้ว่าเป็นเจ้าของ Account ตัวจริงหรือไม่ ถ้าสามารถยืนยันตัวตนได้ก็จะได้รับ OTP เพื่อใช้เป็นรหัสเข้าแอปพลิเคชัน



หน้า Menu



หน้า Take Photo



หน้า Result

หน้า Menu

หน้านี้จะให้ข่าวสารเกี่ยวกับโรคมะเร็งผิวหนังซึ่งผู้ใช้สามารถคลิกเพื่ออ่านข่าวสารเพิ่มเติมได้ สามารถเลือกในส่วนของการถ่ายภาพเพื่อถ่ายรูปในส่วนที่ต้องการทำวินิจฉัยได้ สามารถเลือกดูในส่วนของผลการวินิจฉัยได้ ผู้ใช้ยังสามารถเลือกในส่วนของ Setting เพื่อจัดการกับข้อมูลได้ และยังมี 3 ปุ่มด้านล่าง หน้าปุ่ม Home ปุ่มแจ้งเตือนเพื่อแจ้งข้อมูลข่าวสาร และปุ่มสุดท้ายคือปุ่มโปรไฟล์

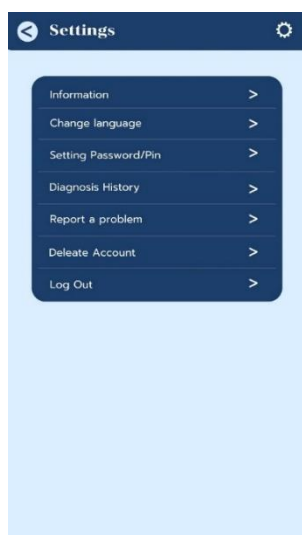
หน้า Take Photo

เป็นหน้าที่ใช้ในการถ่ายรูป เพื่อวินิจฉัยโรคมะเร็งผิวหนังอัตโนมัติ โดยผู้ใช้จะถ่ายภาพของเนื้อเยื่อผิวหนังที่สงสัยว่าเป็นเซลล์มะเร็งผิวหนังหรือไม่ โดยใช้กล้องของโทรศัพท์มือถือเพื่อส่งภาพไปให้ระบบทำการวินิจฉัย

หน้า Result

Result เป็นส่วนที่แสดงผลลัพธ์ที่ถูกวินิจฉัยหลังการถ่ายภาพผิวหนัง ที่ผู้ใช้ได้ทำการถ่ายและทำการแสดงผลลัพธ์ว่าเป็นโรคผิวหนังหรือมะเร็งผิวหนังหรือไม่ หากผิวหนังปกติจะแสดงเป็น Normal skin หากเกิด

โรคผิวหนังหรือมะเร็งผิวหนังจะแสดงชื่อเป็นโรคนี้ๆ และ อธิบายว่าโรคนี้คืออะไร และมีแนวทางในการรักษาอย่างไรต่อไป โดยสามารถกดบันทึกผลลัพธ์ในการวินิจฉัยโรคได้โดยการกดปุ่ม SAVE RESULT และสามารถกดปุ่มย้อนกลับไปหน้าหลักได้ที่ลูกศรมุมซ้ายบน



หน้า Settings



หน้า Information



หน้า Settings

หน้าSettings

เป็นหน้าที่มีตัวเลือกให้ผู้ใช้งานได้เลือกดู คือ ข้อมูล(Information), เปลี่ยนภาษา (Change language), เปลี่ยนรหัส(Setting Password/Pin), ประวัติการวินิจฉัย(Diagnosis History), รายงานปัญหา(Report a problem), ลบบัญชี(Delete Account) และส่วนสุดท้ายคือส่วน Log Out เพื่อออกจากบัญชี

เมื่อเรากดมาที่ Information จะแสดงข้อมูลส่วนบุคคลของผู้ใช้ ประกอบด้วย Personal title (คำนำหน้าชื่อ) โดยสามารถกดปุ่มรูป V เพื่อเลือก คำนำหน้าชื่อ, First Name (ชื่อ), Last Name (นามสกุล), Phone Number (เบอร์โทรศัพท์), Date of Birth (วันเกิดของผู้ใช้) โดยผู้ใช้งานสามารถดูรายละเอียดข้อมูลของตนเองหรือทำการแก้ไข ข้อมูลส่วนบุคคลได้ โดยกดปุ่ม Edit เพื่อทำการแก้ไขข้อมูลและกดปุ่ม ที่ปุ่มเดิมเพื่อทำการsave และสามารถกดปุ่มย้อนกลับไปหน้าหลักได้ที่ลูกศรมุมซ้ายบน

เมื่อเรากดในส่วนของการเปลี่ยนภาษาหน้า Settings จะแสดงส่วนภาษาเพื่อให้ผู้ใช้เลือกเปลี่ยนภาษา

หน้า setting Password Pin

หน้าประวัติการวินิจฉัย

เมื่อเรากดเข้ามาในส่วนของ Setting Password/Pin จะแสดงหน้าต่างของผู้ใช้และผู้ใช้สามารถกดเปลี่ยนรหัสของตนเองได้

เมื่อเรากดเข้ามาในส่วนของหน้า Diagnosis History จะแสดงในส่วนของหน้าประวัติการวินิจฉัยโรคที่ผู้ใช้เคยถ่ายรูปวินิจฉัย

หน้า Report Problem

หน้า Delete account

เมื่อเรากดเข้ามาในส่วนของ Report a problem จะขึ้นในส่วนของ comment เพื่อให้ผู้ใช้พิมพ์รายงานปัญหาที่เกิดขึ้น และสามารถแนบรูปในส่วนที่เกิดปัญหาของแอปได้

เมื่อกดเข้าไปในส่วนที่ Delete account จะขึ้นในส่วนของตัวเลือก yes กับ no เพื่อให้ผู้ใช้ได้เลือก
ถ้าผู้ใช้เลือก yes บัญชีจะถูกลบ แต่ถ้าเลือก no ก็จะกลับไปหน้าจอ Settings

บทที่ 4

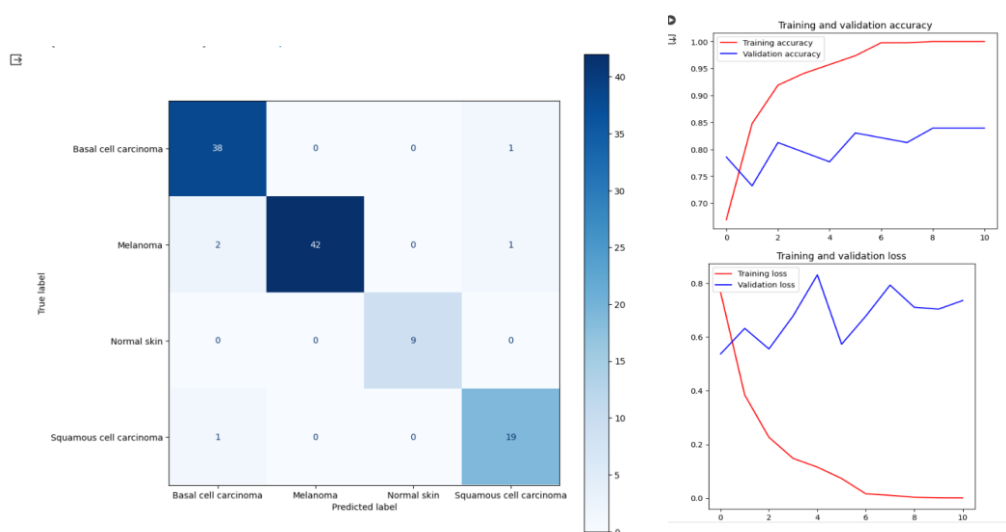
ผลลัพธ์การดำเนินการ/การอภิปรายผลลัพธ์

ผลลัพธ์ประสิทธิภาพ Test accuracy ทั้งก่อนและหลังปรับภาพด้วย DHR ของทั้ง 4 โมเดลสรุปเป็นตารางดังนี้

Model ก่อนการปรับภาพ Test accuracy		Model หลังการปรับภาพ Test accuracy	
EfficientNet V2L	0.9557521939277649	EfficientNet V2L	0.9292035102844238
MobileNetV3small	0.9203540086746216	MobileNetV3small	0.9203540086746216
NASNetLarge	0.8053097128868103	NASNetLarge	0.769911527633667
RestNet-50	0.9292035102844238	RestNet-50	0.9115044474601746

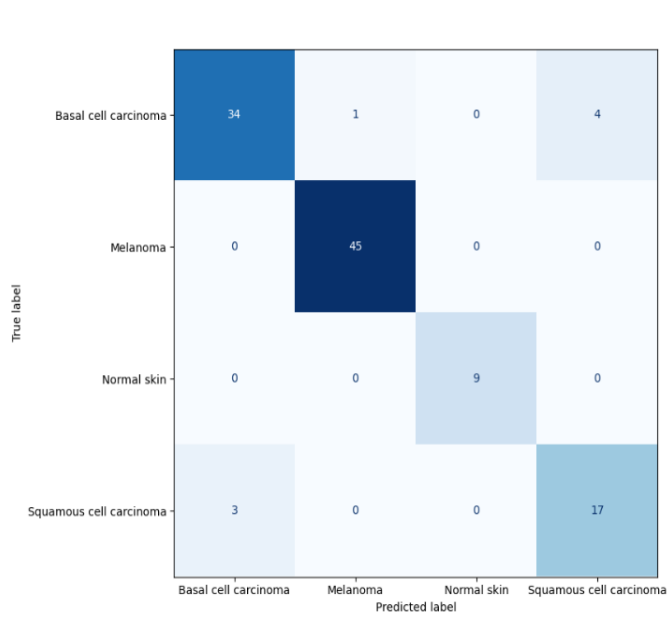
กลุ่มของเราได้เลือกใช้ EfficientNet V2L เนื่องจากเป็นโมเดลที่มีความสมดุลระหว่างประสิทธิภาพและทรัพยากรที่ใช้ อีกทั้งสามารถวินิจฉัยภาพที่ซับซ้อนและภาพใหม่ๆ ได้ดีกว่า ซึ่งมี Test accuracy ของก่อนปรับภาพและหลังปรับภาพอยู่ที่ 0.9557521939277649 และ 0.9292035102844238 ตามลำดับ โดย Test accuracy คือความแม่นยำของโมเดล (model) ในการทำนาย (prediction) หรือการจำแนกประเภทของข้อมูลที่ไม่ได้ใช้ในการฝึก (training) โมเดล นั่นคือ ข้อมูลที่โมเดลไม่เคยเห็นมาก่อน ซึ่งใช้ในการทดสอบ (testing) ความแม่นยำจะถูกคำนวณโดยการเปรียบเทียบค่าที่โมเดลทำนายกับค่าจริงของข้อมูลที่ใช้ทดสอบ โดยปกติแล้วความแม่นยำจะถูกนับเป็นเปอร์เซ็นต์ของจำนวนที่ทำนายถูกต้องเทียบกับจำนวนทั้งหมดของข้อมูลในชุดทดสอบ และมี weighted avg ของ f1-score อยู่ที่ 0.96 (ก่อนปรับภาพ) และ 0.93 (หลังปรับภาพ) ซึ่ง weighted avg คือ ค่าเฉลี่ยเลขคณิตถ่วงน้ำหนัก โดยน้ำหนักของข้อมูลคือจำนวน support หรือจำนวนรูปภาพทั้งหมดในส่วน test

EfficientNet V2L ก่อนลบขนด้วย DHR



จากผลลัพธ์ที่แสดงรูปตารางดังภาพด้านบน ในแนวแกน y คือกลุ่มข้อมูลที่ถูกต้อง และในแนวแกน x คือชื่อของกลุ่มข้อมูลที่โมเดลทำนาย หากแนวแกน y และแกน x ตรงกันแสดงว่าผลทำนายถูกต้อง ตัวเลขในตารางแสดงถึงจำนวนของรูปภาพ dataset ดังนั้นตัวเลขยังคงมีการกระจายอยู่บ้าง หมายความว่ายังคงมีภาพที่โมเดลยังคงทำนายผิดอยู่เล็กน้อย

EfficientNet V2L หลังลบขนด้วย DHR



ในกลุ่มของภาพหลังลบขน ถึงแม้จะมีความผิดพลาดในชุด Basal cell carcinoma และ Squamous cell carcinoma แต่มีการกระจายไปยังกลุ่มอื่นน้อย หมายความว่าหากทำการวินิจฉัย กลุ่มที่มักจะมีวินิจฉัยผิดจะเป็นกลุ่มโรคสองกลุ่มนี้ แต่กลุ่มอื่นจะวินิจฉัยได้อย่างถูกต้องโดยส่วนใหญ่

รายละเอียดเปรียบเทียบ

```
<ipython-input-17-0897c54d69da>:3: UserWarning:
test_loss, test_acc = model_0.evaluate_generat
Test loss: 0.1454925686120987
Test accuracy: 0.9557521939277649
Test Time: 0:00:01.499643
```

```
<ipython-input-17-0897c54d69da>:3: UserWarning:
test_loss, test_acc = model_0.evaluate_generat
Test loss: 0.25416186451911926
Test accuracy: 0.9292035102844238
Test Time: 0:00:01.226464
```

	precision	recall	f1-score	support
Basal cell carcinoma	0.93	0.97	0.95	39
Melanoma	1.00	0.93	0.97	45
Normal skin	1.00	1.00	1.00	9
Squamous cell carcinoma	0.90	0.95	0.93	20
accuracy			0.96	113
macro avg	0.96	0.96	0.96	113
weighted avg	0.96	0.96	0.96	113

	precision	recall	f1-score	support
Basal cell carcinoma	0.92	0.87	0.89	39
Melanoma	0.98	1.00	0.99	45
Normal skin	1.00	1.00	1.00	9
Squamous cell carcinoma	0.81	0.85	0.83	20
accuracy			0.93	113
macro avg	0.93	0.93	0.93	113
weighted avg	0.93	0.93	0.93	113

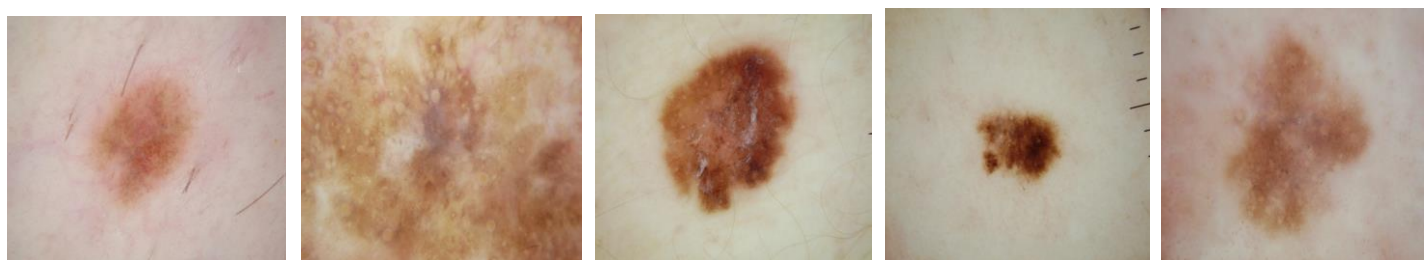
ดังนั้นกลุ่มของเราจึงเลือกโมเดล EfficientNet V2L แบบที่มีการปรับภาพลบขนออกด้วย DHR เนื่องจากมีการกระจายข้อมูลที่ผิดน้อยกว่า ถึงแม้ว่ารายละเอียดของ Test accuracy ของกลุ่มที่ไม่ปรับภาพจะ

มีค่าสูงกว่า เพื่อให้แน่ใจทางกลุ่มจึงได้ทำการทดสอบผลการรันของโมเดลทุกโมเดล โดยการสุ่มภาพตัวอย่างจาก valid มา 5 ภาพ และรันวัดผลประสิทธิภาพด้วยชุดภาพเดียวกัน

สุ่มภาพตัวอย่างวัดประสิทธิภาพการทำงานของโมเดลที่เลือก

ทางกลุ่มได้สุ่มตัวอย่างออกมา 5 ภาพจากแต่ละกลุ่มโรค สำหรับการวัดผลประสิทธิภาพของโมเดล EfficientNet V2L ทั้ง 2 รูปแบบ โดยภาพที่สุ่มมีดังนี้

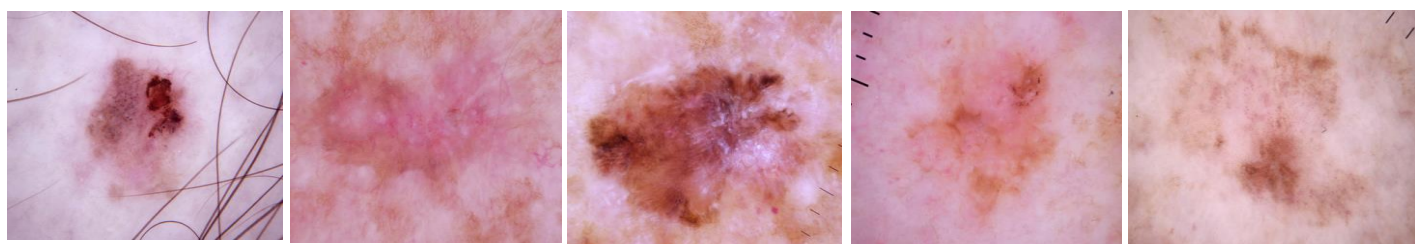
MELANOMA



BASAL CELL CARCINOMA



SQUAMOUS CELL CARCINOMA



NORMAL SKIN



ผลความถูกต้องจากการสุ่มตัวอย่างมา 5 ภาพที่เหมือนกัน

Disease (ก่อนปรับ)	EfficientNet V2L	MobileNetV3 small	NASNetLarge	RestNet-50	Disease (หลังปรับ)	EfficientNet V2L	MobileNetV3 small	NASNetLarge	RestNet-50
Basal cell carcinoma	4/5	3/5	4/5	4/5	Basal cell carcinoma	5/5	4/5	5/5	4/5
Squamous cell carcinoma	4/5	3/5	2/5	4/5	Squamous cell carcinoma	5/5	5/5	2/5	4/5
Melanoma	5/5	5/5	5/5	5/5	Melanoma	5/5	5/5	5/5	5/5
Normal skin	5/5	5/5	4/5	5/5	Normal skin	5/5	5/5	5/5	5/5

จากผลลัพธ์ที่แสดงดังตาราง จะเห็นได้ว่าส่วนใหญ่การรันของชุดข้อมูลแบบปรับภาพ จะมีผลการรันที่ดีกว่าผลการรันของชุดข้อมูลที่ไม่ปรับภาพ โดยผลการรันที่ดีที่สุด คือ EfficientNet V2L ของชุดข้อมูลที่มีการปรับภาพลบจน เมื่อได้โมเดลที่มีประสิทธิภาพที่น่าพอใจ กลุ่มเราจะทำการปรับพารามิเตอร์ด้วยโมเดลที่เลือกต่อ เพื่อให้การวินิจฉัยมีประสิทธิภาพสูงสุด จากงานวิจัยที่ได้ศึกษา

หลังจากที่ได้ปรับพารามิเตอร์

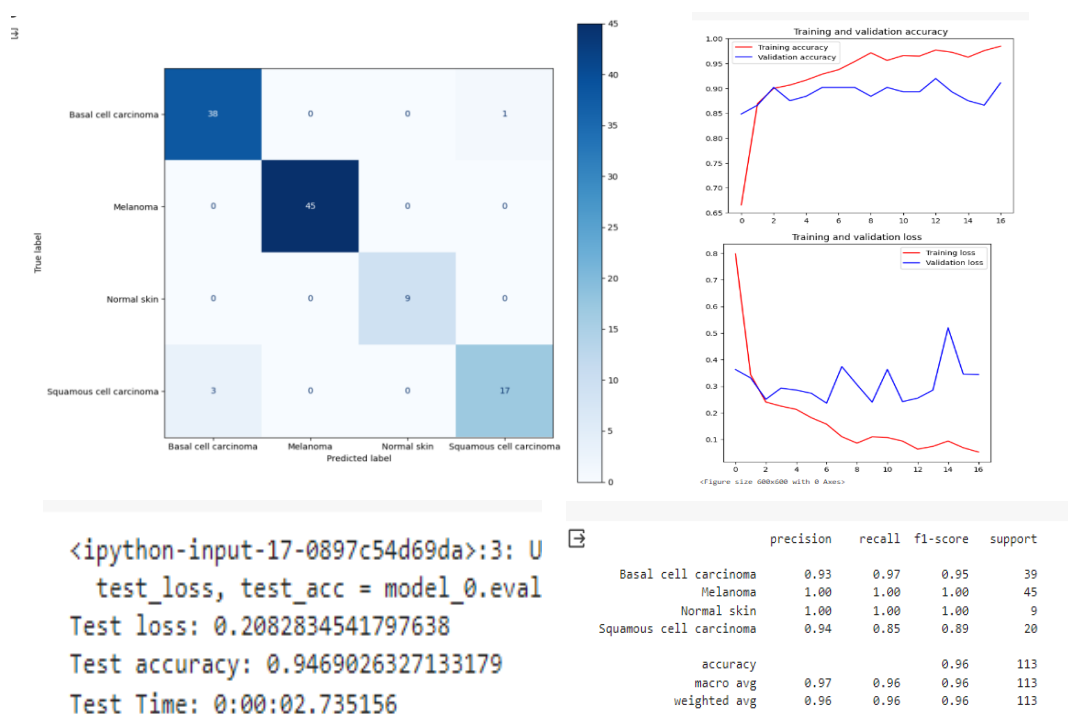
ผลลัพธ์เปรียบเทียบจากการปรับพารามิเตอร์ ด้วยกลุ่ม Dataset เดียวกันคือมีการปรับภาพลบจนด้วย DHR

ก่อนปรับพารามิเตอร์	Test accuracy	หลังปรับพารามิเตอร์	Test accuracy
EfficientNet V2L	0.9292035102844238	EfficientNet V2L	0.9469026327133179
MobileNetV3small	0.9203540086746216	MobileNetV3small	0.9292035102844238
NASNetLarge	0.769911527633667	NASNetLarge	0.8584070801734924
RestNet-50	0.9115044474601746	RestNet-50	0.8672566413879395

เปรียบเทียบผลการรันจากการสุ่มภาพตัวอย่างด้วยกลุ่ม Dataset เดียวกันคือมีการปรับภาพลบชนด้วย DHR

Disease (ก่อนปรับ)	EfficientNet V2L (ก่อนการปรับพารามิเตอร์)	EfficientNet V2L (หลังการปรับพารามิเตอร์)
Basal cell carcinoma	4/5	4/5
Squamous cell carcinoma	4/5	4/5
Melanoma	5/5	5/5
Normal skin	5/5	5/5

ผลลัพธ์โมเดลในรูปแบบกราฟและรายละเอียดหลังจากปรับพารามิเตอร์



จากผลลัพธ์ทั้งหมดที่แสดงดังด้านบนของการปรับพารามิเตอร์ ก่อนการปรับ EfficientNet V2L มีค่า Test accuracy เท่ากับ 0.9292035102844238 และหลังการปรับมีค่าอยู่ที่ 0.9469026327133179 ซึ่งมีค่าเพิ่มจากเดิม ทางกลุ่มจึงดูประสิทธิภาพผลการรันต่อ เพื่อให้แน่ใจว่ามีผลที่ดีขึ้นจริงๆ ในตารางเปรียบเทียบผลการรัน พบว่ามีประสิทธิภาพเท่าๆกัน ดังนั้นจึงตรวจสอบที่กราฟ พบว่ามีการกระจายข้อมูลที่ผิดพลาดลง โดยในกลุ่มที่มักมีความผิดพลาดอย่าง Basal cell carcinoma และ Squamous cell carcinoma ก็มีความผิดพลาดที่น้อยลงเมื่อเทียบกับก่อนการปรับพารามิเตอร์ในกลุ่มภาพที่ลบชน ดังนั้นกลุ่มเราจึงได้โมเดลที่มีประสิทธิภาพที่ดีที่สุดคือ EfficientNet V2L ด้วยชุด dataset ที่มีการปรับแต่งลบชนด้วย DHR และมีการปรับพารามิเตอร์

บทสรุปและแนวทางแนะนำ

โดยสรุปแล้วกลุ่มของเราเลือกโมเดล EfficientNet V2L เนื่องจากมีผลลัพธ์ที่มีประสิทธิภาพมากที่สุดจาก Test accuracy และเลือกรูปแบบที่มีการปรับพารามิเตอร์เนื่องจากการกระจายข้อมูลเมทริกซ์ดีกว่า ส่วน dataset เลือกใช้ในกลุ่มที่มีการปรับแต่งภาพจาก DHR ซึ่งได้ผลลัพธ์ประสิทธิภาพการรันดีที่สุดโดยดูจากตารางการสุมภาพ และกลุ่มเรามีการออกแบบ UX/UI ที่ใช้งานสะดวก เข้าใจง่าย สีที่ใช้มีความสบายต่อดวงตา และตัวแอปพลิเคชันเป็นภาษาไทย สามารถใช้ได้ทุกเพศทุกวัย มีฟังก์ชัน ดังนี้

- ถ่ายภาพเพื่อตรวจเช็คสภาพผิวหนัง โดย
- แสดงผลลัพธ์ว่าเป็นโรคผิวหนังหรือมะเร็งผิวหนังหรือไม่
- หากผิวหนังปกติจะแสดงเป็น Normal skin
- หากเกิดโรคผิวหนังหรือมะเร็งผิวหนังจะแสดงชื่อเป็นโรคนั้นๆ
- รายละเอียดโรคเบื้องต้นและมีแนวทางในการรักษาอย่างไรต่อไป
- สามารถกดบันทึกผลลัพธ์ในการวินิจฉัยโรคได้
- สามารถติดตามข่าวสารผ่านตัวแอปพลิเคชัน
- สามารถพิสูจน์ตัวตน เพื่อป้องกันความปลอดภัยของการโจรกรรมข้อมูล และไม่เป็นการเผยแพร่ความลับของผู้ใช้งาน

และในอนาคตจะมีการพัฒนาในส่วนของโค้ดให้ดีขึ้นมากกว่าเดิมและมีความแม่นยำโดยใช้เครื่องมือต่างๆใน Library ที่นอกเหนือจากปัจจุบัน เช่น เทคนิค Focal Loss และปรับพารามิเตอร์ over fit พร้อมทั้งต่อยอดนำไปทำเป็นแอปต่อ โดยจะพัฒนาระบบด้วย Android SDK บนสมาร์ตโฟนที่มีระบบปฏิบัติการแอนดรอยด์ 4.2 ขึ้นไป เพื่อให้สามารถรองรับ Google map API version 2 และใช้ Eclipse Development Tools and Java Development Kit (JDK) ในส่วนของ Web-based Application และใช้ CMS และ Google map API version 3 เชื่อมโยงกับฐานข้อมูลของระบบแอปพลิเคชันของกลุ่มเรา

อุปสรรคและปัญหาที่พบ

เนื่องจากการลบขนออกจากภาพทุกภาพ และการ train AI ทั้ง 4 โมเดลโดยใช้ภาพก่อนลบขนและหลังลบขน ต้องใช้ทรัพยากร GPU จำนวนมาก แต่ google colab มีการจำกัด GPU ที่ใช้ได้ เมื่อใช้หมดก็จะเปลี่ยนไปใช้ CPU ซึ่งการประมวลผลด้วย CPU จะช้ากว่ามาก และยังมีจำกัด ทำให้ CPU เต็มก่อนที่จะประมวลผลเสร็จ ทำให้ต้องประมวลผลใหม่ ซึ่งทำให้ต้องใช้เวลามาก วิธีแก้ปัญหาคือ ใช้ account google colab หลาย account ในการรัน เพื่อให้มี GPU ที่ใช้ได้มากขึ้น และใช้โค้ดเพื่อปรับขนาดของรูปก่อนที่จะลบขนเพื่อลดการใช้ GPU และลดเวลาที่ใช้

บรรณานุกรม

- ณัฐรฐนนท์ กานต์วีกุลธนา. (19 ตุลาคม 2562). *Overfitting* กับการแก้ปัญหา. เข้าถึงได้จาก Medium: <https://medium.com/@natratanonkanraweekultana/overfitting-กับการแก้ปัญหา-7eed084c17f3>
- สุรพงศ์ กนกทิพย์สถาพร. (11 พฤษภาคม 2563). *AI วินิจฉัยโรคมะเร็งผิวหนัง 7 ชนิด ความแม่นยำ 94% Melanoma Skin Cancer HAM10000 Dermoscopic Pigmented Lesions – Image Classification ep.8*. เข้าถึงได้จาก BUA Labs: <https://www.bualabs.com/archives/4122/ai-diagnose-melanoma-skin-cancer-mnist-ham10000-dermatoscopic-pigmented-lesions-image-classification-ep-8/>
- พิชดา สายเชื้อ. (2565). *การจำแนกรูปภาพจอประสาทตาที่มีภาวะเบาหวานขึ้นตาด้วยการเรียนรู้เชิงลึก*. ภัทรมน พันธุ์แพง และ สัญญา พันธุ์แพง. (2563). *แอปพลิเคชันวิเคราะห์ภาพถ่ายจากอุปกรณ์ตรวจเท้าด้วยตนเอง สำหรับผู้ป่วยเบาหวาน*.
- ปิยะวัฒน์ หนูเล็ก และ กรวิทย์ พฤษชัยนิมิต ขวัญกมล ดิฐกัญจน์. (2565). *วินิจฉัยมะเร็งผิวหนังจากภาพถ่ายโทรศัพท์ด้วยโครงข่ายประสาทเทียมสังวัตนาการ*. (ม.ป.ป.).

ภาพผนวก ก
เครื่องมือที่ใช้

1. ใช้ google colab ในการรันโค้ด
2. ใช้โค้ด Digital Hair Removal (DHR) ในการลบขนออกจากรูป

```
# DHR ลบเส้นขนออกจากกรุปใน train_path

# กำหนดเส้นทางไปยังโฟลเดอร์ที่มีรูปภาพ
folder_path = train_path
output_folder = DHR_train_path

# กำหนดขนาดสูงสุดของไฟล์ภาพ (500 KB)
max_file_size = 500 * 1024 # ใ้หน่วยไบต์

# วนซ้ำทุกไฟล์ในโฟลเดอร์
for root, dirs, files in os.walk(folder_path):
    for file_name in files:
        # ตรวจสอบว่าไฟล์เป็นรูปภาพหรือไม่
        if file_name.endswith(".jpg") or file_name.endswith(".png"):
            full_file_path = os.path.join(root, file_name)
            src = cv2.imread(full_file_path)

            # ลดขนาดรูปภาพถ้าไฟล์มีขนาดเกินที่กำหนด
            file_size = os.path.getsize(full_file_path)
            if file_size > max_file_size:
                scale = (max_file_size / file_size) ** 0.5
                new_size = (int(src.shape[1] * scale), int(src.shape[0] * scale))
                src = cv2.resize(src, new_size, interpolation = cv2.INTER_AREA)
```

```
# Convert the original image to grayscale
grayScale = cv2.cvtColor(src, cv2.COLOR_RGB2GRAY)

# Kernel for the morphological filtering
kernel = cv2.getStructuringElement(1,(17,17))

# Perform the blackhat filtering on the grayscale image to find the hair countours
blackhat = cv2.morphologyEx(grayScale, cv2.MORPH_BLACKHAT, kernel)

# intensify the hair countours in preparation for the inpainting algorithm
ret,thresh2 = cv2.threshold(blackhat,10,255,cv2.THRESH_BINARY)

# inpaint the original image depending on the mask
dst = cv2.inpaint(src,thresh2,1,cv2.INPAINT_TELEA)

# บันทึกภาพลงในโฟลเดอร์ย่อย
output_file_path = os.path.join(output_folder, "processed_" + file_name)
cv2.imwrite(output_file_path, dst)
```