โครงงานวิทยาการคอมพิวเตอร์

ภาควิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

มหาวิทยาลัยเกษตรศาสตร์

เรื่อง NihonGO!

คณะผู้จัดทำโครงงาน

นายศิริสุข ทานธรรม

นายปฏิภัทธิ์ อุคำ

นายธนอนันท์ เฉลิมพันธ์

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 01418364

การเรียนรู้เชิงลึกในทางปฏิบัติ Practical Deep Learning

ภาคเรียนที่ 2 ประจำปีการศึกษา 2566

โครงงานวิทยาการคอมพิวเตอร์

ภาควิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

มหาวิทยาลัยเกษตรศาสตร์

เรื่อง NihonGO!

คณะผู้จัดทำโครงงาน

นายศิริสุข ทานธรรม 6610402230

นายปฏิภัทธิ์ อุคำ 6610402141

นายธนอนันท์ เฉลิมพันธ์ 6610402078

อาจารย์ที่ปรึกษาโครงการ

ผศ.ดร.ชาคริต วัชโรภาส

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 01418364

การเรียนรู้เชิงลึกในทางปฏิบัติ Practical Deep Learning

ภาคเรียนที่ 2 ประจำปีการศึกษา 2566

บทคัดย่อ

NihonGO! มีจุดประสงค์เพื่อรู้จำอักษรคันจิจากลายมือเขียน (Hand-writing kanji recognition) จำนวน 81 ตัวอักษรอ้างอิงจากการสอบวัดระดับภาษาญี่ปุ่น JLPT N5 โดยใช้ชุดข้อมูลจาก สถาบันวิทยาศาสตร์และเทคโนโลยีอุตสาหกรรมขั้นสูงแห่งชาติของญี่ปุ่น (AIST) และเก็บข้อมูลเพิ่มเติมด้วยตนเอง โดยใช้เครื่องมือที่ใช้ในการศึกษาด้วยหลักการของ Convolutional Neural Network (CNN) ในการพัฒนาโมเดลในรูปแบบของการวิเคราะห์ รูปภาพ (Image classification) โดยใช้ LeNet-5 และ VGG16 โดยทำความสะอาดชุดข้อมูล (Image pre-processing) โดยใช้หลักการของ Binary thresholding โดยมีการวัดผลเป็นในรูปของ Confusion Matrix, Graph Loss , Accuracy และ F1-Scores ผลจากการศึกษาพบว่าโมเดล LeNet-5 ที่ใช้โครงสร้างของ LeNet-5 สามารถทำนายผลตัวอักษรคันจิแบบเขียนได้ดีกว่าโมเดล VGG16 มากกว่าที่ค่าความแม่นยำ 0.01 %

สารบัญ

บทที่ 1
บทนำ

ที่มาและความสำคัญของโครงงาน

ภาษาญี่ปุ่นเป็นภาษาที่ได้รับความนิยมเป็นอย่างมากจากชาวต่างชาติเช่นเดียวกันกับกับภาษาอังกฤษ และภาษาจีน อย่างไรก็ตาม ตัวอักษรในญี่ปุ่นมีลักษณะของระบบตัวอักษรที่มีความซับซ้อนทั้งในเรื่องของรูปร่างต่าง ๆ ที่เรียกว่าคันจิ ซึ่งทำให้ในการทำการสกัดตัวอักษร OCR (Optical Character Recognition) มีความท้าทาย และยากในการแยกแยะตัวอักษรที่มีความคล้ายคลึงกันในตัวอักษรคันจิ ที่ใช้ในชีวิตประจำวันในการอ่านเขียน หรือ แปลเอกสารเป็นต้น ผู้พัฒนาจึงได้สร้างโปรเจค NihonGO! ขึ้นมาเพื่อศึกษาความเป็นไปได้ในการใช้หลักการของ Convolutional Neural Network (CNN) ในรูปแบบของการคัดแยกตัวอักษร (Image Classification) ในการรู้จำประเภทของตัวอักษรคันจิแบบเขียนมือ เพื่อที่จะนำไปพัฒนา และปรับใช้กับระบบ OCR ต่อในอนาคต

จุดประสงค์ของการศึกษาค้นคว้า

- เพื่อนำไปประยุกต์ใช้เป็นระบบทำนายตัวอักษรคันจิ auto-correction จากตัวอักษรเขียนมือจากผู้ใช้งานที่เป็น hand writing
- เพื่อศึกษาประสิทธิภาพของการใช้เทคนิคของ Convolutional Neural Network ไปใช้ในการจำแนกประเภทของตัวอักษรคันจิ ในรูปแบบของการจำแนกรูปภาพ (Image classification)

สมมติฐานของการศึกษาค้นคว้า

- โมเดลสามารถตรวจจับตัวอักษรคันจิได้ แม้จะมีลักษณะที่ใกล้เคียงมากก็ตาม

ขอบเขตของการศึกษาค้นคว้า

- ทดลองโดยเทคนิคโมเดล Convolutional Neural Network โดยการใช้ VGG16 และ LeNet-5 ในการ fine-tuning เพื่อพัฒนาโมเดลขึ้นมาใหม่ ในลักษณะของการจำแนกรูปภาพ
- ข้อมูลตัวอักษรคันจิจำนวน 81 ตัวอักษรในระดับของการสอบวัดระดับภาษาญี่ปุ่น JLPT ระดับ N5 ที่มีความหลากหลายในเรื่องของลักษณะ และรูปร่างของคันจิ

นิยามศัพท์

- **OCR (Optical Character Recognition)** : การรู้จำอักขระด้วยแสง (OCR) เป็นกระบวนการที่แปลงภาพข้อความให้เป็นรูปแบบข้อความที่เครื่องอ่านได้ ตัวอย่างเช่น หากคุณสแกนแบบฟอร์มหรือใบเสร็จ คอมพิวเตอร์ของคุณจะบันทึกการสแกนดังกล่าวเป็นไฟล์รูปภาพ คุณไม่สามารถใช้ตัวแก้ไขข้อความเพื่อแก้ไข ค้นหา หรือนับคำในไฟล์รูปภาพได้ อย่างไรก็ตาม คุณสามารถใช้ OCR เพื่อแปลงรูปภาพเป็นเอกสารข้อความที่มีการจัดเก็บเนื้อหาเป็นข้อมูลตัวอักษรได้

- **fine-tuning** : เป็นกระบวนการในการฝึก (training) โมเดลปัญญาประดิษฐ์ที่มีการถูกสร้างมาก่อนแล้ว (pre-trained model) โดยปรับปรุงและปรับเปลี่ยนโมเดลเหล่านั้นให้สอดคล้องกับงานหรือข้อมูลที่ต้องการใช้งาน เราสามารถทำ fine-tuning โมเดลโดยการฝึกโมเดลบนชุดข้อมูลของเราเองหลังจากที่มันถูกฝึกด้วยชุดข้อมูลใหญ่ ๆ ที่มีอยู่แล้ว

- **JLPT-N5** : คือระดับภาษาญี่ปุ่นของ Japanese-Language Proficiency Test (JLPT) หรือที่รู้จักกันในชื่อว่า "N5" ซึ่งเป็นระดับที่ต่ำที่สุดในประเภทของการทดสอบความสามารถในการใช้ภาษาญี่ปุ่นเพื่อการสื่อสารทั่วไป ในเรื่องของการอ่าน การเขียน การฟัง และการพูด ระดับนี้มักเรียกว่า "Beginner" หรือระดับเริ่มต้น

ประโยชน์ที่คาดว่าจะได้รับ

- เป็นแนวทางในการพัฒนาต่อยอดต่อในเทคนิคอื่น ๆ เช่น ในลักษณะของการทำ OCR (Optical Character Recognition) ในการวิเคราะห์ในลักษณะของประโยค
- ต่อยอดไปในด้านการนำไปใช้งาน หรือเป็นกรณีในการศึกษา เช่น การประยุกต์ใช้ในงานด้านการแปลภาษาต่อไป

บทที่ 2
เอกสารที่เกี่ยวข้อง

ตอนที่ 1 ทฤษฎีหลักการ และเทคโนโลยีที่ใช้พัฒนาโมเดล

1.1) Neural Network คือเทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ที่จำลองการเรียนรู้ของสมองมนุษย์ ผ่านโครงข่ายประเทียมเป็น Node หลายตัวใน 1 layer และเชื่อมต่อกันแบบสมบูรณ์ (Fully Connected) ที่แต่ละโหนด มี Function ทางคณิตศาสตร์ในการเรียนรู้ เรียกว่า Activation Function เพื่อใช้ในการทำนายค่าออกมา และปรับค่า weight ของ Function ด้วยเทคนิค gradient descent และ backpropagation

1.2) Gradient decent เป็นเทคนิคที่ใช้ในการปรับ weight ของ function โดยดูจากผลต่างของ loss function ของค่าที่โมเดลทำนายได้ และค่าจริงเพื่อหา global minimum หรือจุดต่ำสุดที่ weight ทำให้โมเดลมีความไกล้เคียงกับค่าจริงมากที่สุด

1.3) Backpropagation เป็นเทคนิคที่ใช้ปรับค่า weight ของโมเดลที่ได้จากการหาจากเทคนิค gradient decent ย้อนกลับจาก layer สุดท้ายของโมเดล

1.4) Optimizer เป็นคือ algorithms ที่ใช้หาวิธีหรือเส้นทางที่ไกล้ที่สุดในการคำนวณของเทคนิค gradient decent และ backpropagation ในการคำนวณทางคณิตศาสตร์เพื่อปรับค่า weight ของ parameter แต่ละ node และลดค่า loss function

1.4.1) Adam : เป็น algorithm optimizer ที่เทคนิคในการรับ weight ของพารามิเตอร์ แบบปรับตามสถานการณ์ (dynamic)

1.5) Activation Function เป็น function ทางคณิตศาสตร์ใน node ที่ใช้เรียนรู้ในการเรียนรู้แต่ละครั้งของโมเดล

1.5.1) ReLu เป็น activation function ที่ให้ผลลัพท์เป็นจำนวนจริงบวก

1.5.2) Softmax เป็น activation function ที่ให้ผลลัพท์เป็น percent

1.6) Batch Size เป็นการกำหนดค่าจำนวนข้อมูลในการเรียนรู้ของโมเดลในหนึ่งครั้ง

1.7) Epoch เป็นหน่วยวัดการเรียนรู้ของโมเดลเมื่อเรียนรู้ด้วยจำนวน batch size ทั้งหมดเท่ากับจำนวข้อมูลที่ใช้เรียนรู้ 1 ครั้ง

1.8) Normalization เป็นการปรับค่าชุดข้อมูลให้เป็นค่าเชิงตัวเลขที่คอมพิวต์สามารถเข้าใจได้หรือ ปรับให้เป็นช่วงตัวเลขที่เหมาะกับการประมวลผล

1.9) Regularization เป็นการปรับโมลเดลไม่ให้มีการเรียนรู้ให้ยึดติดกับชุดข้อมูลที่ใช้เรียนรู้มากเกินไป(overfit) เพื่อให้สามารถทำนายข้อมูลทั่วไปได้ถูกต้อง เช่น Dropout และ EarlyStopping เป็นต้น

1.10) Convolutional เป็นเทคนิคในการลดขนาด parameter ของรูปภาพเรื่องจากขนาดรูปภาพที่มากขึ้นทำให้มีจำนวนพารามิตเตอร์ในการเรียนรู้ของโมเดลและมีระยะเวลาในการเรียนรู้ที่แปรผันตามจำนวนพารามิเตอร์ จึงใช้เทคนิค convolutional โดยใช้ kernel ตรวจจับ feature ของ image ผ่าน max pooling function
ให้ได้เฉพาะชุดพารามิเตอร์ที่เกี่ยวข้อง

1.11) Max Pooling เป็นการลดขนาดของข้อมูลรูปภาพเช่น 2x2

1.12) TensorFlow เป็น library พัฒนาขึ้นโดย Google สำหรับ สร้างโมแดล Deep Learning ผ่านการเรียกใช้ APIs เพื่อช่วยจัดการโครงสร้างและกำหนดค่าพารามิเตอร์

1.13) Keras เป็น ส่วนปรับปรุง library ของ TensorFlow ที่ช่วยทำให้สามารถเรียกใช้ APIs ของ TensorFlow ได้ง่ายขึ้น ซึ่งสามารถเรียกใช้ได้ผ่าน library TensorFlow

1.14) Sequential Model เป็น function ใน Keras library ที่ช่วยในการสร้างโมเดลการเรียนรู้ให้เป็นรูปแบบลำดับ

1.15) การตั้ง Random seed เป็นการกำหนดให้ค่าตัวแปรสุ่มภายในโปรแกรมมีค่าคงที่เสมอเพื่อให้สามารถวัดประสิทธิภาพในการเรียนรู้แม่นยำ

1.16) Augmentation เป็นแปลงชุดข้อมูล (datasets) ประเภทรูปภาพให้ได้ชุดข้อมูลเดิมที่มีความละเอียดมากยิ่งขึ้นเช่น ปรับภาพให้เอียง การกลับด้าน หรือการหมุนเป็นต้น

1.17) Keras Image Generator เป็น function ใน library ของ keras ที่ช่วยให้สามารถทำImage Augmentation จากข้อมูลจริง ได้ง่ายขึ้น

ตอนที่ 2 ทฤษฎีหลักการ และเทคโนโลยีที่ใช้วัดประสิทธิภาพ

2.1) การ Train เป็นช่วงที่มีกระบวนการในการเรียนเพื่อปรับ weight ของ model ให้เรียนรู้และจดจำข้อมูลได้ โดยใช้ชุดข้อมูล เดียวกับ Validation โดยมี ratio มากกว่า

2.2) การ Validation เป็นช่วงที่กระบวนการทดสอบประสิทธิภาพการเรียนรู้ของโมเดล ที่ใช้ทดสอบประสิทธิภาพกับชุดข้อมูล เดียวกันกับ Train แต่มี ratio น้อยกว่า

2.3) การ Test เป็นช่วงที่กระบวนการทดสอบประสิทธิภาพการเรียนรู้ของโมเดล ที่ใช้ทดสอบประสิทธิภาพกับชุดข้อมูลใหม่แยกออกจาก Train และ Validation เพื่อวัดประสิทธิภาพจริงของ model

2.4) Categorical Crossentropy คือ ค่าเฉลี่ยของ Cross-Entropy ที่เกิดจากการแจกแจงความน่าจะเป็น 2 แบบ คือ การแจกแจงความน่าจะเป็นที่เราอยากได้ (Actual) กับการแจกแจงความน่าจะเป็นที่ถูกประมาณโดย Model (Predicted) ของ Class ต่างๆ ซึ่งการได้ค่าเฉลี่ยน้อยดีกว่าการได้ค่าเฉลี่ยมาก

2.5) Transfer Learning คือการนำโมเดลที่มีการเรียนรู้ weight จาก ชุดข้อมูลขนาดใหญ่หรือมีประสิทธิภาพกับชุดข้อมูลที่สอดคล้องกับที่ต้องการ เพื่อลดระยะเวลาในการเทรนและพิ่มประสิทธิภาพโมเดล

2.6) LeNet5 คือ โมเดล Deep Learning ที่เรียนรู้จากข้อมูลรูปภาพ MNIST ใช้เทคนิค Convolution เพื่อลดขนาด parameter และเพิ่มประสิทธิภาพ ด้วยโครงสร้าง

2.6.1) Conv2D 6 kernel 5x5

2.6.2) MaxPooling2D 2x2

2.6.3) Conv2D 16 kernel 5x5

2.6.4) MaxPooling2D 2x2

2.6.5) Flatten

2.6.6) Dense 120

2.6.7) Dense 84

2.6.8) Output layer

2.7) VGG16 คือโมเดล Deep Learning ที่มี 16 layers convolution และ dense layers ที่เรียนรู้จากข้อมูล ImageNet RGB โดยใช้หลักการ convolution 3x3 และ relu ต่อกันหลาย layers มีพารามิเตอร์ทั้งหมด 138 ล้านพารามิเตอร์

2.8) classification reports เป็นการแสดงข้อมูล report ของโมเดล

2.8.1) loss validation เป็นการแสดงค่าความผิดพลาดของโมเดล ค่า loss น้อยบ่งบอกถึงประสิทธิภาพดี

2.8.2) accuracy คือการวัดประสิทธิภาพเฉลี่ยเป็น percent จาก (True Positive + True Negative / total predictions)

2.8.3) precision เป็นค่าวัดอัตราส่วนของตัวอย่างที่โมเดลคาดการณ์ว่าเป็นคลาส X ที่เป็นคลาส X จริง จาก (TP/TP+FP) ค่า precision ที่สูงบ่งบอกว่าโมเดลมี False Positive น้อย

2.8.4) recall วัดอัตราส่วนของตัวอย่างที่เป็นคลาส X จริงๆ ที่โมเดลคาดการณ์ว่าเป็นคลาส X (TP/TP+FN) ค่า recall ที่สูงบ่งบอกว่าโมเดลมี False Negative น้อย

2.8.5) f1-score ค่าเฉลี่ยถ่วงน้ำหนักระหว่าง precision และ recall (2 *(precision *recall/precision+recall))

2.8.6) confusion matrix แสดงจำนวนตัวอย่างที่โมเดลคาดการณ์ว่าเป็นคลาสใดเทียบกับคลาสที่แท้จริง True Possitive, True Negative, False Possitive, False Negative

บทที่ 3
อุปกรณ์และวิธีการดำเนินงาน

1.) อุปกรณ์และโปรแกรมที่ใช้

1.1) อุปกรณ์

1) คอมพิวเตอร์/Laptop

1.2) โปรแกรม

1) Visual Studio Code

2) TensorFlow Keras

3) OpenCV2

2.) ขั้นตอนการดำเนินงาน

2.1) การศึกษาและค้นคว้าข้อมูล

- จากการหาข้อมูลเกี่ยวกับตัวอักษรคันจิที่มีจากอินเตอร์เน็ต และ จากหน่วยงานภาครัฐเช่น NECTEC พบว่าข้อมูลส่วนใหญ่จะเป็นภาพอักษรคันจิแบบเขียนโบราณ หรือจากลายพู่กัน ซึ่งไม่เหมาะกับการที่จะใช้ในการทำนายตัวอักษรคันจิ เนื่องด้วยลักษณะของลายเส้น และรูปแบบการเขียนที่แตกต่างกันอย่างมีนัยสำคัญ ทางผู้พัฒนาจึงได้ทำการสืบหาข้อมูลเพิ่มเติม และขอคำปรึกษาจากนักวิจัยใน NECTEC เลยได้ชุดข้อมูลจากทาง AIST ที่เป็นชุดข้อมูลคันจิที่เก็บตัวอย่างจากลายมือเขียนคันจิแบบปกติในช่วงปี 1973 - 1984 ซึ่งมีความใกล้เคียงกับลายมือเขียนคันจิในยุคปัจจุบัน ผู้พัฒนาจึงได้เลือกใช้ข้อมูลชุดนี้จาก AIST

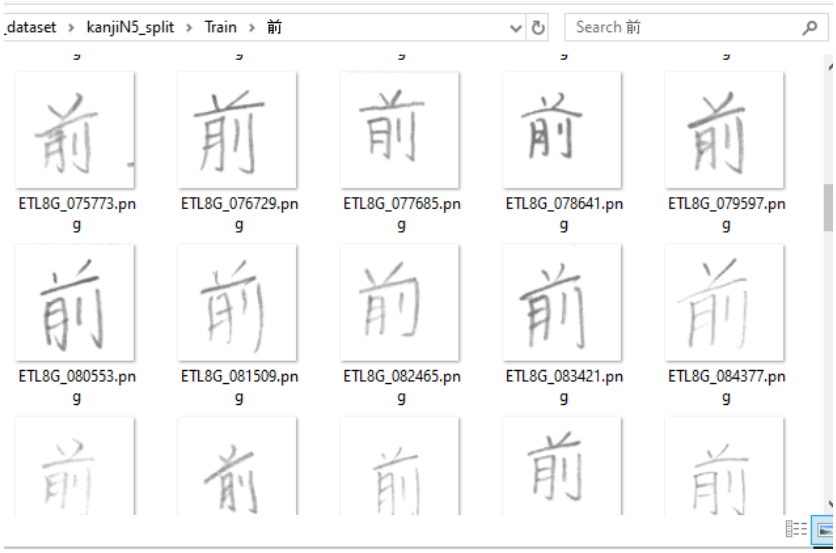
จากภาพชุดตัวอักษรจากด้านบนเป็นลักษณะของอักษรคันจิเขียนโบราณที่ใช้พู่กัน
ส่วนภาพด้านล่างเป็นตัวอักษรคันจิที่ใช้เขียนปกติในปัจจุบัน

2.2) การเก็บรวบรวมข้อมูล

ในส่วนนี้แสดงถึงวิธีการเก็บข้อมูลสำหรับการฝึกโมเดลเป็นขั้นตอนในการพัฒนาและประเมินโมเดล เราต้องการให้ข้อมูลทั้งจำนวนในชุดข้อมูลฝึกและชุดข้อมูลทดสอบมีความสมมาตร เพื่อให้สามารถวิเคราะห์ผลลัพธ์ได้อย่างถูกต้อง โดยสามารถทำได้ดังขั้นตอนต่อไปนี้

2.2.1) การเก็บรวบรวมข้อมูล

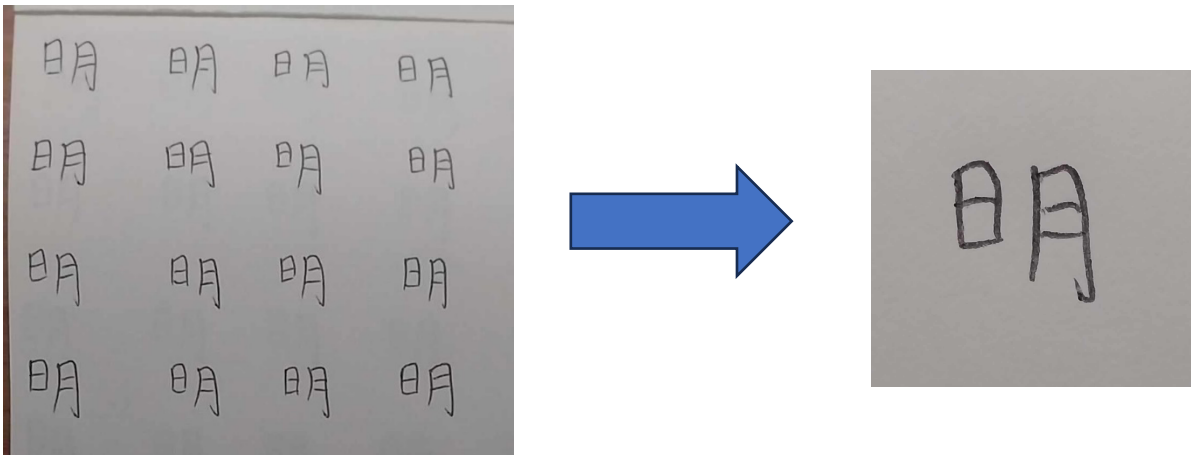
ข้อมูลประเภทตัวอักษรคันจิได้มาจาก ETL Character Database จากสถาบันวิทยาศาสตร์ และเทคโนโลยีอุตสาหกรรมขั้นสูงแห่งชาติของญี่ปุ่น AIST โดยทำการขอ datasets จากทาง AIST ผ่านทาง email etlcdb (aist.go.jp)



ลักษณะ datasets เป็นตัวอักษรคันจิที่เป็นลายมือเขียนจำนวน 957 ตัวอักษร และในแต่ละ labels มีจำนวน 160 ภาพ

2.2.2) การเก็บข้อมูลเพิ่มเติม

ทำการเก็บข้อมูลเพิ่มเติมโดย การเลือกสุ่มคันจิมาหนึ่งตัวจากใน Website Shirabe jisho 1 labels แล้วทำการเขียนด้วยมือ จำนวน 160 ตัวอักษร แล้วบันทึกเป็นรูปภาพเพื่อใช้รวบรวมกับ datasets หลัก



ทำการแปลงข้อมูลดิบเป็นข้อมูลที่ใช้ในการเทรนโมเดล

2.2.3) การทำความสะอาดข้อมูล (Image processing)

ในส่วนนี้จะทำให้ข้อมูลรูปภาพมีความคมชัดมากขึ้นกว่าเดิม โดยใช้หลักการของ binary thresholding ในการทำ Image Processing

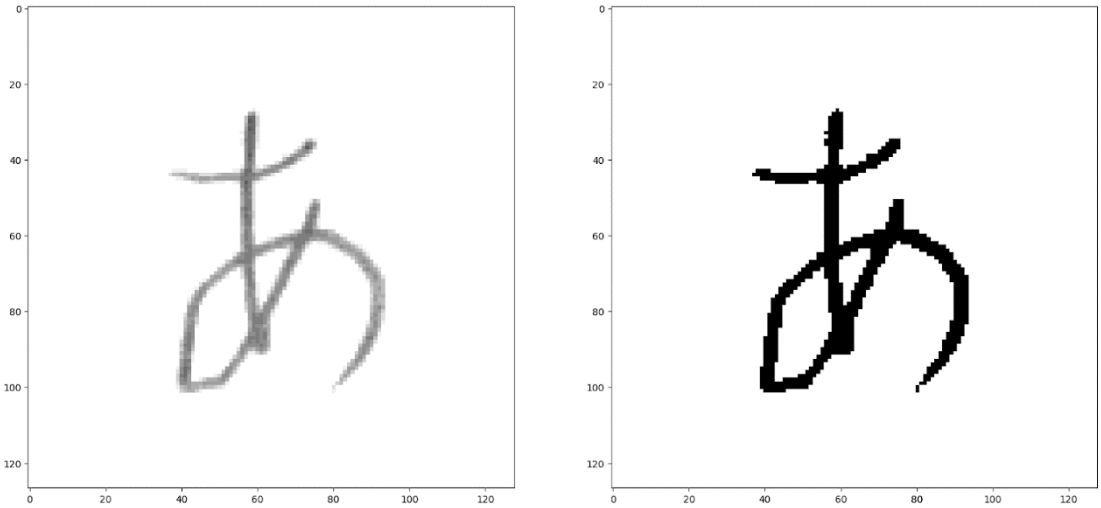
```
import cv2
import numpy as np
image1 = cv2.imread(img_path)
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)

plt.figure(figsize=(20, 20))
plt.subplot(1, 2, 1)
plt.imshow(image1, 'gray')

plt.subplot(1, 2, 2)
plt.imshow(thresh1, 'gray')
cv2.imwrite('./a.png', thresh1)
```

จากภาพด้านบนทำการใช้ฟังก์ชัน cv2.threshold() โดยแบ่งค่า pixel เป็นแบบ binary โดยกลุ่มแรกคือ ค่ามากกว่า threshold ที่กำหนดไว้ กับกลุ่มที่สอง คือ ค่าน้อยกว่าหรือเท่ากับค่า threshold ที่กำหนดไว้โดย cv2.THRESH__BINARY ในการ

1. ใช้ในการกำหนดค่าที่น้อยกว่าหรือเท่ากับ threshold ในที่นี้ก็คือ 120 ให้เป็น 0
2. ค่าที่มากกว่า threshold หรือ 120 ให้เป็น max value



ภาพที่ได้หลังจากการทำ binary thresholding ด้วย OpenCV

3.) การทำ Data Preprocessing ในการแบ่งข้อมูล

ในส่วนนี้จะแสดงถึงวิธีการการเตรียมข้อมูลสำหรับเพื่อ Training โมเดล และปรับแต่งการเรียนรู้ของโมเดล ด้วยการเพิ่มความหลากหลายให้กับชุดข้อมูลของเราผ่านการเพิ่มข้อมูล (data augmentation) และการจัดเรียงชุดข้อมูล (data shuffling) สำหรับการฝึกและทดสอบโมเดล

```
data_augmentation = tf.keras.preprocessing.image.ImageDataGenerator(
    height_shift_range=0.1,
    width_shift_range=0.1,
    zoom_range=0.1,
)
```

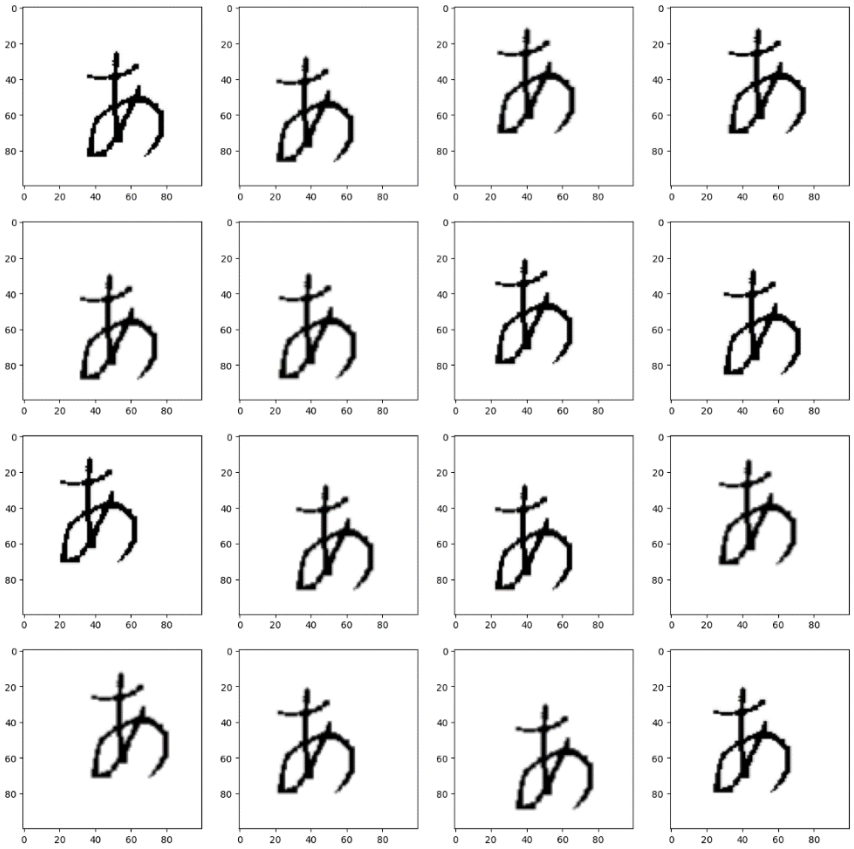
โค้ดด้านบนที่กำหนด data_augmentation เป็นการใช้งานคลาส ImageDataGenerator ใน TensorFlow ซึ่งเป็นเครื่องมือสำหรับการปรับแต่งภาพในรูปแบบต่างๆ เพื่อเพิ่มความหลากหลายให้กับชุดข้อมูลภาพ โดยในที่นี้มีการกำหนดพารามิเตอร์ต่างๆ เพื่อกำหนดการปรับแต่งของภาพดังนี้:

3.3.1) height_shift_range: กำหนดช่วงการเคลื่อนที่แนวตั้งของภาพเป็นเปอร์เซ็นต์ของความสูงของภาพ ในที่นี้กำหนดเป็น 0.1 ซึ่งหมายถึงภาพสามารถเคลื่อนที่ขึ้นหรือลงได้ไม่เกิน 10% ของความสูงของภาพต้นฉบับ

3.3.2) width_shift_range: กำหนดช่วงการเคลื่อนที่แนวนอนของภาพเป็นเปอร์เซ็นต์ของความกว้างของภาพ ในที่นี้กำหนดเป็น 0.1 ซึ่งหมายถึงภาพสามารถเคลื่อนที่ไปทางซ้ายหรือ ขวาได้ไม่เกิน 10% ของความกว้างของภาพต้นฉบับ

3.3.3) zoom_range: กำหนดช่วงการซูมของภาพเป็นเปอร์เซ็นต์ของการซูมเข้าหรือซูมออก ในที่นี้กำหนดเป็น 0.1 ซึ่งหมายถึงภาพสามารถซูมเข้าหรือซูมออกได้ไม่เกิน 10% ของขนาดภาพต้นฉบับ

data_augmentation ที่กำหนดขึ้นนี้จะใช้สร้างภาพที่มีการเคลื่อนที่และการซูมเข้า-ออกเล็กน้อยเพื่อเพิ่มความหลากหลายให้กับชุดข้อมูลภาพในระหว่างการฝึกโมเดลการเรียนรู้ของเครื่อง ทำให้โมเดลสามารถเรียนรู้และจำแนกภาพได้อย่างถูกต้องแม้แต่ในเงื่อนไขที่ภาพมีความแตกต่างเล็กน้อย



ตัวอย่างภาพที่ได้จากการทำ Image augmentation

4.) การสร้างชุดข้อมูล

4.4.1) การสร้างชุดข้อมูลฝึก (Training Dataset)

```
train_ds = data_augmentation.flow_from_directory(
    directory=os.path.join(data, "Train"),
    class_mode='categorical',
    batch_size=batch_size,
    target_size=(image_size, image_size),
    shuffle=True
)
```

เราใช้ flow_from_directory เพื่อโหลดข้อมูลจากโฟลเดอร์ที่กำหนด ซึ่งในที่นี้คือโฟลเดอร์ "Train" ที่มีภาพที่เราจะใช้ในการฝึกโมเดล เรากำหนด class_mode เป็น 'categorical' เนื่องจากเรามีหลายๆ กลุ่มของคลาส (หรือหมวดหมู่) และเราต้องการให้โมเดลทำการจำแนกแต่ละคลาส เรากำหนด batch_size เพื่อระบุจำนวนข้อมูลที่จะถูกโหลดในแต่ละรอบ และ target_size เพื่อระบุขนาดของภาพที่เราต้องการให้มีหลังจากการปรับขนาด เรากำหนด shuffle=True เพื่อให้ข้อมูลถูกสับเปลี่ยนในแต่ละรอบของการฝึก เพื่อปรับให้โมเดลไม่ถูกสร้างความคาดหวังจากลำดับของข้อมูล เพื่อลดการ Overfitting

4.4.2) การสร้างชุดข้อมูลทดสอบ (Validation Dataset)

```
valid_ds = data_augmentation.flow_from_directory(
    directory=os.path.join(data, "Valid"),
    class_mode='categorical',
    batch_size=batch_size,
    target_size=(image_size, image_size),
    shuffle=False
)
```

เหมือนกับการสร้างชุดข้อมูลฝึก เราใช้ flow_from_directory เพื่อโหลดชุดข้อมูลทดสอบจากโฟลเดอร์ที่ระบุ ในที่นี้คือโฟลเดอร์ "Valid" ซึ่งมีภาพที่เราจะใช้ในการทดสอบความแม่นยำของโมเดล เรากำหนด class_mode เป็น 'categorical' เหมือนกับการสร้างชุดข้อมูลฝึก เรากำหนด shuffle=False เพื่อให้ข้อมูลไม่ถูกสับเปลี่ยน เนื่องจากเราต้องการทราบผลลัพธ์จากการทดสอบโมเดลที่มีชุดข้อมูลเดิม

4.4.3) การสร้างชุดข้อมูลทดสอบ (Test Dataset)

```
test_ds = data_augmentation.flow_from_directory(
    directory=os.path.join(data, "Test"),
    class_mode='categorical',
    batch_size=batch_size,
    target_size=(image_size, image_size),
    shuffle=False
)
```

การสร้างชุดข้อมูลทดสอบเหมือนกับการสร้างชุดข้อมูลทดสอบ โดยเราใช้ flow_from_directory เพื่อโหลดชุดข้อมูลทดสอบจากโฟลเดอร์ที่ระบุ ในที่นี้คือโฟลเดอร์ "Test" ซึ่งมีภาพที่เราจะใช้ในการทดสอบความแม่นยำของโมเดล เรากำหนด class_mode เป็น 'categorical' เหมือนกับการสร้างชุดข้อมูลฝึก เรากำหนด shuffle=False เพื่อให้ข้อมูลไม่ถูกสับเปลี่ยน เนื่องจากเราต้องการทราบผลลัพธ์จากการทดสอบโมเดลที่มีชุดข้อมูลเดิม

5.) ขั้นตอนการเลือกประเภท และการเทรนโมเดล

ในขั้นตอนนี้เราจะเลือกใช้ 2 โมเดลในการทดลองซึ่งก็คือ LeNet-5 และ VGG16 เนื่องจากเป็นโมเดลที่นิยมใช้ในการทำ CNN และทั้งสองโมเดลมีขนาด parameters ที่แตกต่างกันอย่างเห็นได้ชัด

5.1) การเลือกประเภทของโมเดล

5.1.1) VGG16

- VGG16 ซึ่งเป็นโมเดลที่มีโครงสร้าง CNN ที่ถูกพัฒนาโดย (Karen Simonyan) และ (Andrew Zisserman) สำหรับการจำแนกประเภทของภาพ ที่ประกอบด้วยมีพารามิเตอร์จำนวนมากและมีการฝึกด้วยชุดข้อมูล ซึ่งประกอบด้วยภาพขนาดใหญ่ที่มีหลายๆ หมวดหมู่ การใช้โมเดลที่ถูกฝึกด้วยข้อมูลจำนวนมาก โมเดลจะสามารถให้ความความแม่นยำในการจำแนกหมวดหมู่ของภาพได้เป็นอย่างดี

5.2.2) LeNet-5

- LeNet-5 เป็นโมเดลที่มีโครงสร้าง CNNที่ถูกพัฒนาขึ้นโดยญาน์ ลูแคน (Yann LeCun) และเพื่อนร่วมทีมในปี 1998 ซึ่งประกอบด้วยนักวิจัยอย่างเลอองบอตตู (Léon Bottou) โยชัว บังกิโอ (Yoshua Bengio) และแพทริก แฮฟเนอร์ (Patrick Haffner) โดยมีวัตถุประสงค์เพื่อใช้ในการจำแนกและจดจำอักษรบนซอฟต์แวร์โทรสับคอมพิวเตอร์ของบริษัทแอทแอนด์ทีแทต (AT&T) ในประเทศสหรัฐอเมริกา โดยที่ LeNet-5 เป็นโมเดลแรกที่ใช้เทคนิคการทำคอนโวลูชันแบบลอยเลื่อน (convolutional layer) ร่วมกับการลดขนาดของข้อมูลด้วยการทำ max-pooling

5.2) การสร้างโมเดล

5.1.1) VGG16

```
base_model = VGG16(
    weights='imagenet',
    include_top=False,
    input_shape=(image_size, image_size, 3)
)

base_model.trainable = False
```

จากโค้ดเป็นโมเดลที่ถูกโหลดมาจาก VGG16 โดยใช้คำสั่ง VGG16() การระบุ weights='imagenet' หมายถึงการโหลดน้ำหนักที่ถูกฝึกไว้แล้ว include_top=False หมายถึงไม่รวมชั้น Fully Connected Layer ด้านบนสุดของโมเดลคำสั่ง base_model.trainable = False ใช้ในการตั้งค่าโมเดลเบส (base_model) ให้ไม่สามารถเรียนรู้ (trainable) ได้ นั่นหมายความว่าในขั้นตอนการเทรนโมเดลหลัก (base_model) จะไม่มีการปรับปรุงค่าของน้ำหนัก (weights) ของชั้น (layers) ภายในโมเดลในระหว่างกระบวนการฝึกโมเดลทั้งหมด ดังนั้นค่าของน้ำหนักที่ถูกโหลดมาจากโมเดล VGG16 ซึ่งฝึกด้วยชุดข้อมูล ImageNet จะไม่ถูกเปลี่ยนแปลงในขั้นตอนการฝึกโมเดลใหม่ เพื่อให้รักษาความแม่นยำและคุณสมบัติของโมเดลที่ถูกฝึกมาแล้ว และป้องกันการที่ค่าของน้ำหนักจะถูกเปลี่ยนแปลงจากชุดข้อมูลใหม่ที่ไม่ได้เป็นเหมือนกับชุดข้อมูลที่ใช้ในการฝึกเดิม

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)
```

จากนั้นสร้างโมเดลที่ประกอบด้วยโมเดล Pre-Train (base model) และชั้นที่เราเพิ่มเติมเข้าไปและกำหนด Output เป็น Class ที่เรากำหนด

- x = base_model.output เราเริ่มด้วยการนำเอา output ที่ได้จากโมเดล Pre-Train (base model) ซึ่งเป็น output ของชั้นสุดท้ายของโมเดล VGG16 หรือชั้นที่เรียกว่า "block5_pool" มาใช้งานต่อในโมเดลที่เรากำลังสร้าง

- x = GlobalAveragePooling2D()(x) เราได้ใช้ใช้ Global Average Pooling 2D เพื่อลดขนาดของ feature maps ที่ได้จากโมเดล Pre_Train เป็นเวกเตอร์เดียว โดยการเอาค่าเฉลี่ยของทุกๆ feature map ในแต่ละชั้น ซึ่งช่วยลดจำนวนพารามิเตอร์และความซับซ้อนของโมเดลได้

- x = Dense(1024, activation='relu')(x) เพิ่มชั้น fully connected layer ที่มี 1024 units โดยใช้ activation function เป็น ReLU (Rectified Linear Unit) เพื่อให้โมเดลเรียนรู้และปรับค่าในระหว่างการฝึก

- x = Dropout(0.5)(x) เพิ่มชั้น Dropout เพื่อลดโอกาสการเกิด overfitting โดยการสุ่มปิดบางช่อง (units) ของชั้นก่อนขั้นตอนการคำนวณถัดไปในการฝึก ในที่นี้กำหนดให้ปิดช่องออกไป 50% (0.5)

- predictions = Dense(num_classes, activation='softmax')(x) เพิ่มชั้น fully connected layer สุดท้ายที่มีจำนวน units เท่ากับจำนวนของคลาสทั้งหมดที่เราต้องการจำแนก ซึ่งในที่นี้กำหนดให้เป็น num_classes = 81 ซึ่งคำนวณโดยใช้ activation function เป็น softmax เพื่อให้ได้ผลลัพธ์ที่เป็นความน่าจะเป็นของแต่ละคลาส

- model = Model(inputs=base_model.input, outputs=predictions) สร้างโมเดลทั้งหมดโดยระบุ inputs เป็น input ของโมเดลเบส และ outputs เป็น output ของโมเดลที่เราได้ปรับแก้และเพิ่มเติมเข้าไป

5.1.2) LeNet-5

```
model = Sequential([
    Conv2D(6, (5,5), input_shape=(image_size,
image_size, 3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(16, (5,5), activation='relu'),
    MaxPooling2D(2,2),
    Dropout(0.25),
    Flatten(),
    Dense(120, activation='relu'),
    Dense(84, activation='relu'),
    Dense(num_classes, activation='softmax')
])
```

จากโค้ดเป็นโมเดล CNN ที่กำหนดขึ้นมานี้เป็นโมเดลที่ใช้ในการจำแนกหมวดหมู่ของภาพซึ่งได้อ้างอิงมาจากโมเดล LeNet-5 โดยมีลักษณะโครงสร้างดังนี้

- **Convolutional Layers (Conv2D):** โมเดลนี้เริ่มต้นด้วยชั้น Convolutional Layer ที่มี 6 filters และ kernel size เป็น (5,5) โดยกำหนด activation function เป็น ReLU (Rectified Linear Unit) เพื่อให้เกิดการกระตุ้นเซลล์ที่เป็นบวกในการเรียนรู้คุณลักษณะของภาพ
- **MaxPooling Layers (MaxPooling2D):** หลังจาก Convolutional Layer แต่ละชั้น มีการนำเอา MaxPooling Layer มาใช้เพื่อลดขนาดของภาพและลดความซับซ้อนของโมเดล ทำให้การคำนวณเร็วขึ้นและลดการ overfitting
- **Dropout** ในที่นี้กำหนดค่าเป็น 0.25 ซึ่งเป็นการป้องกันการ overfitting โดยการสุ่มที่จะปิดบางเซลล์ในชั้นก่อนถึงชั้นที่ต่อมาในระหว่างการฝึก
- **หลังจากการใช้ MaxPooling** ในชั้นที่สอง โมเดลจะมีการ Flatten เพื่อแปลงข้อมูลจากฟีเจอร์มาพร้อมสำหรับการนำเข้าในชั้น Dense
- **มี Dense Layer** สองชั้น ซึ่งเป็นชั้น Fully Connected Layer หลังจาก Flatten Layer โดยมีจำนวนโหนดเป็น 120 และ 84 ตามลำดับ กำหนด activation function เป็น ReLU

สุดท้ายคือชั้น Output Layers(Dense) สำหรับการแยกประเภทของภาพ โดยมีจำนวนโหนดเท่ากับจำนวนคลาสที่ต้องการจำแนก และกำหนด activation function เป็น softmax เพื่อให้ได้ผลลัพธ์ที่เป็นความน่าจะเป็นของการจำแนกแต่ละคลาส

5.3) การเทรนโมเดล

5.3.1) Callback

```
filepath = "./new_labels_weight/save-{epoch:02d}-{loss:.4f}.h5"
vggpatience = 3
lenetpatience = 10

callback = [
    EarlyStopping(patience=patience, monitor='val_accuracy'),
    ModelCheckpoint(filepath, monitor='val_accuracy',
                    verbose=1, save_best_only=True,
                    mode='max')
]
```

ในการเทรนโมเดล เราใช้ Callbacks เพื่อตรวจสอบและควบคุมกระบวนการเทรน ในตัวอย่างนี้มีการใช้ EarlyStopping ซึ่งจะหยุดกระบวนการเทรนเมื่อไม่มีการปรับปรุงความแม่นยำบนชุดการตรวจสอบ (validation set) เป็นเวลา 3 รอบ สำหรับโมเดล Fine-Tune จาก VGG 16 และ 10 รอบ สำหรับโมเดล LeNet-5 และ ModelCheckpoint ซึ่งจะบันทึกโมเดลที่ดีที่สุดเมื่อมีความแม่นยำบนชุดการตรวจสอบ

5.3.2) การ Compile Model

```
model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=1e-6),
    loss=keras.losses.CategoricalCrossentropy(from_logits=False),
    metrics=['accuracy'])
```

Optimizer จากโค้ดนี้ใช้ optimizer เป็น Adam ซึ่งเป็นวิธีการปรับค่า learning rate แบบอัตโนมัติตาม Gradient ของ gradient descent เพื่อช่วยในการควบคุมการเรียนรู้ของโมเดล Loss Function ใช้เป็น Categorical Crossentropy เป็นฟังก์ชันสูญเสียที่ใช้ในการประมาณความแตกต่างระหว่างการทำนายจากโมเดลกับค่าเป้าหมาย ในกรณีที่มีหลายคลาส (multi-class classification)

Metrics ในกรณีนี้ใช้ metrics เพียงแค่ accuracy เพื่อวัดประสิทธิภาพของโมเดลในการจำแนกคลาส

5.3.3) Fitting model

```
history = model.fit(
    train_ds,
    epochs=epochs,
    validation_data=valid_ds,
    callbacks=callback)
```

ในส่วนนี้เป็นการฝึกโมเดลโดยใช้ข้อมูลจาก train_ds และทดสอบโมเดลด้วยข้อมูลจาก valid_ds โดยกำหนดจำนวนรอบการฝึก (epochs) เท่ากับ 100 รอบสำหรับ VGG 16 และ 20 รอบสำหรับ LeNet-5 และใช้ข้อมูล validation_data เพื่อตรวจสอบประสิทธิภาพของโมเดลในแต่ละรอบการฝึก

callback ที่ใช้ประกอบด้วย EarlyStopping และ ModelCheckpoint ถูกใช้เพื่อหยุดการฝึกโมเดลหากไม่มีการปรับปรุงผลลัพธ์ (val_accuracy) ภายใน 3 รอบการฝึกสำหรับ VGG 16 และ 10 รอบการฝึกสำหรับ LeNet-5 และบันทึกโมเดลที่ดีที่สุด (ตาม val_accuracy) ไว้ในแต่ละรอบการฝึก

การทำงานในขั้นตอนนี้จะแสดงผลการฝึกโมเดลตามจำนวนรอบที่กำหนด โดยแสดงค่า loss และ accuracy ของข้อมูลฝึกและข้อมูลทดสอบในแต่ละรอบ ซึ่งสามารถใช้ในการประเมินและเปรียบเทียบประสิทธิภาพของโมเดลในแต่ละรอบการฝึก

6) ขั้นตอนการวัดผล

ในส่วนนี้จะแสดงถึงขั้นตอนการวัดผลซึ่งเป็นขั้นตอนสำคัญในการประเมินประสิทธิภาพของโมเดลและการตัดสินใจเกี่ยวกับการปรับปรุงหรือการเลือกใช้โมเดลที่เหมาะสมในงานหรือปัญหาที่กำลังจะแก้ไข ดังนั้น ขั้นตอนการวัดผลจะประกอบไปด้วยขั้นตอนต่อไปนี้

6.1) การวิเคราะห์และวัดผลระหว่างการ Training

การตรวจสอบประสิทธิภาพของโมเดลระหว่างการ Training เป็นขั้นตอนในการช่วยให้โมเดลสามารถปรับ Weight เพื่อให้โมเดลทำงานได้อย่างเหมาะสม โดยการวิเคราะห์และวัดผลระหว่างการฝึกสามารถทำได้โดยใช้ข้อมูล Training Dataset และ Validation Dataset โดยมีหลักการดังต่อไปนี้

6.1.1) การคำนวณค่า Accuracy และ ค่า Loss

ค่า accuracy และ loss จะถูกคำนวณขึ้นมาทุก Epoch ของการฝึก ซึ่งช่วยในการติดตามว่าโมเดลกำลังเรียนรู้อย่างไร ค่า accuracy สูงและค่า loss ต่ำจะแสดงให้เห็นถึงประสิทธิภาพของโมเดลในการจำแนกหรือการทำนายข้อมูล

6.1.2) การใช้ Callback

เราใช้ EarlyStopping และ ModelCheckpoint ช่วยในการควบคุมกระบวนการฝึกโดยอัตโนมัติ โดย EarlyStopping จะหยุดการฝึกเมื่อค่าความแม่นยำบนชุดการตรวจสอบหยุดเพิ่มขึ้น และ ModelCheckpoint จะบันทึกโมเดลที่ดีที่สุดตามค่าความแม่นยำบนชุดการตรวจสอบ

6.2) การวิเคราะห์และวัดผลหลังการ Training

- หลังจากการเทรนโมเดลเสร็จสิ้น สามารถนำผลลัพธ์ที่ได้มาวิเคราะห์และวัดผลเพื่อทำความเข้าใจเกี่ยวกับประสิทธิภาพของโมเดลได้ โดยสามารถใช้การรายงานผลลัพธ์จาก classification_report และ confusion_matrix เพื่อทำการวิเคราะห์ได้ดังนี้

6.1.2) Classification Report

- ใน Classification Report จะประกอบด้วยค่า precision, recall, F1-score และ support สำหรับแต่ละคลาสที่ได้จากการทำนายของโมเดล การวิเคราะห์ classification report ช่วยให้เข้าใจถึงประสิทธิภาพของโมเดลในการจำแนกคลาสแต่ละคลาสอย่างละเอียด

6.1.3) Confusion Matrix

- Confusion matrix เป็นตารางที่แสดงความถูกต้องและความผิดพลาดในการจำแนกคลาส ซึ่งแสดงจำนวนของข้อมูลที่ถูกจำแนกถูกต้องและไม่ถูกต้องในแต่ละคลาส การวิเคราะห์ confusion matrix ช่วยในการระบุว่าโมเดลมีประสิทธิภาพในการจำแนกคลาสใดบ้าง และคลาสใดที่มักจะทำให้โมเดลสับสนบ้าง

บทที่ 4
ผลการดำเนินการโครงงาน

4) การวัดผล และการวิเคราะห์ผลลัพธ์

ในส่วนนี้จะเป็นการแสดงถึงวิธีการวิเคราะห์ผลลัพธ์และวัดผลโมเดลหลังจากการฝึก ซึ่งขั้นตอนที่สำคัญในการประเมินประสิทธิภาพของโมเดลที่สร้างขึ้น เพื่อพิสูดว่าโมเดลที่ได้สร้างขึ้นมานั้น จะสามารถนำไปใช้งานได้หรือไม่

4.1) ตัวแปรที่ใช้ในการวัดผล

- Train accuracy, Validation accuracy, Test accuracy
- Train loss, Validation loss, Test loss
- Confusion-Matrix

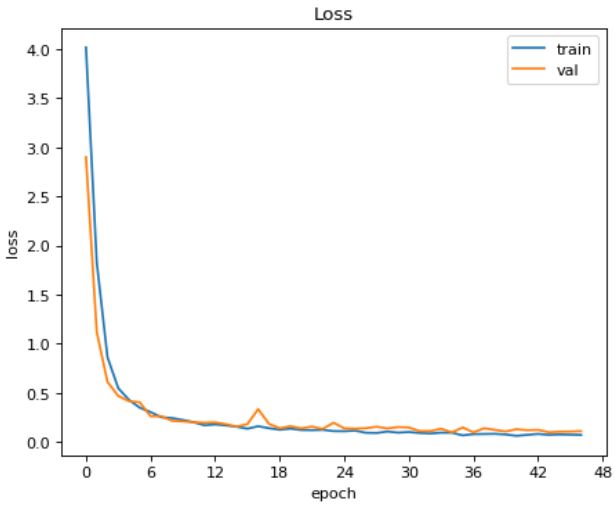
4.2) วัดผลโมเดลที่ได้จาก LeNet-5 และ VGG16

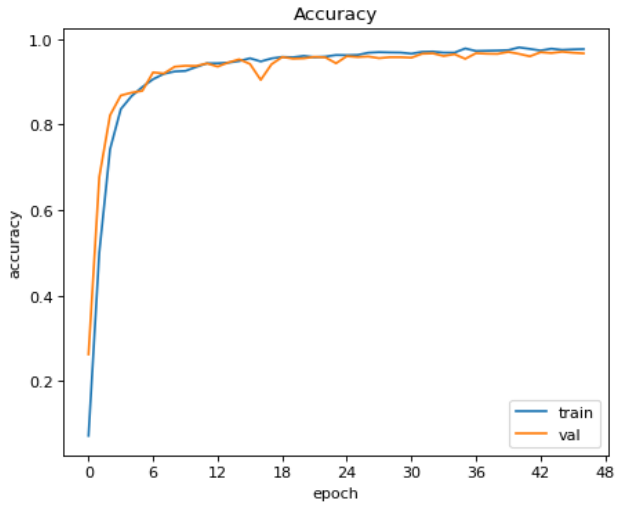
- จากการเทรนโมเดลมีการใช้ Early Stopping โดย monitoring ค่าของ val_loss โดยกำหนดค่า patience เท่ากับ 3 ทั้ง 2 โมเดล

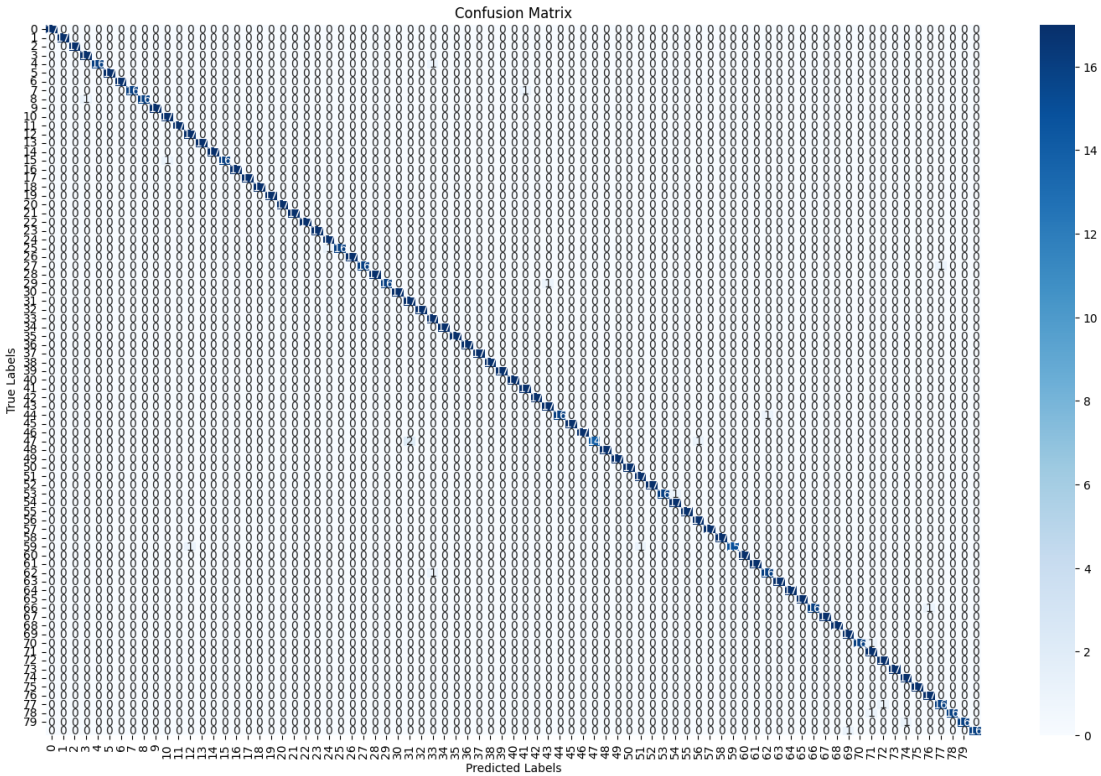
4.2.1) LeNet-5

- จากการเทรนโมเดลโดยใช้ค่า Dropout เท่ากับ 0.25 พบ พบว่า ค่า average training accuracy เท่ากับ 0.98 % , ค่า average validation accuracy เท่ากับ 0.96 % , ค่า training loss เท่ากับ 0.03

และค่า validation loss เท่ากับ 0.09 มากไปกว่านั้นค่า average testing accuracy เท่ากับ 0.992 % และค่า testing loss เท่ากับ 0.022

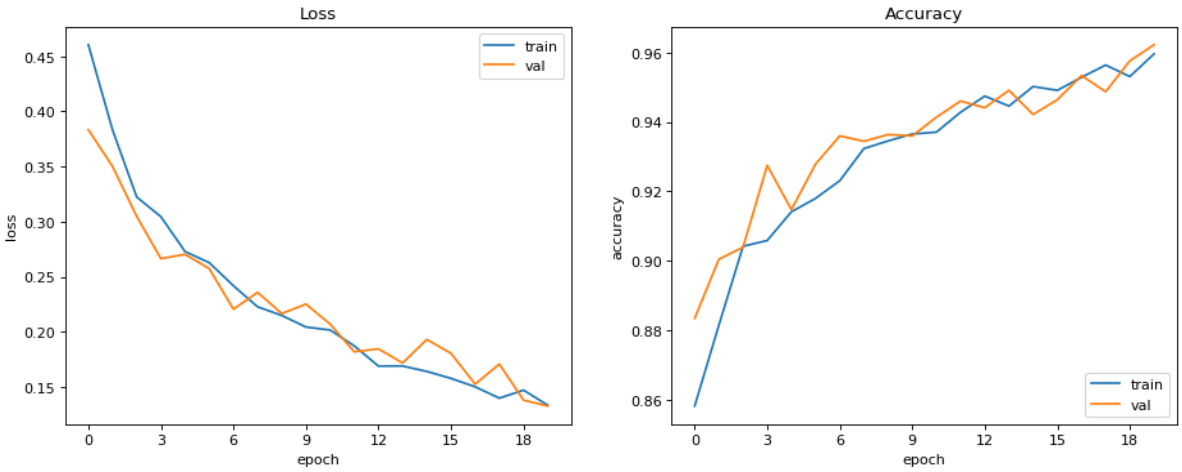


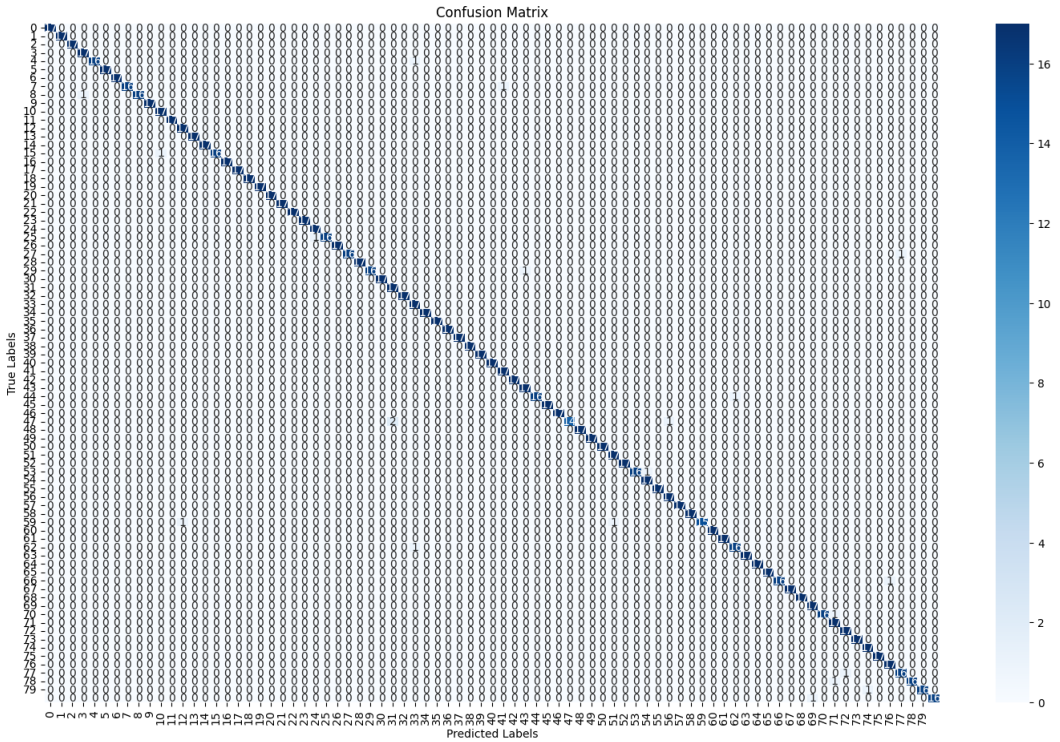




4.2.2) VGG16 (fine tuning 15 layers with ImageNet)

- จากการเทรนโมเดลโดยใช้ค่า Dropout เท่ากับ 0.5 พบว่าค่า average training accuracy เท่ากับ 0.98 % , ค่า average validation accuracy เท่ากับ 0.95 % , ค่า training loss เท่ากับ 0.05 และค่า validation loss เท่ากับ 0.16 มากไปกว่านั้นค่า average testing accuracy เท่ากับ 0.98 % และค่า testing loss เท่ากับ 0.05





4.3) ทดสอบกับข้อมูลจากลายมือเขียนจริงผ่าน Streamlit ที่ deploy บน Huggingface space

- ทำการทดลองโดยการเขียนอักษรคันจิผ่าน Sketchpad tools ใน streamlit ที่มีลักษณะเหมือนการเขียนผ่านกระดาษโดย จำนวนทั้งหมด 5 ตัวอักษรคันจิจำนวน 10 ครั้ง ต่อ 1 ตัวอักษร แล้วให้ model ทำนายคลาส

ตัวอย่างคันจิที่เลือกใช้ในการทดลองที่มีลักษณะที่ใกล้เคียงกัน

1. 木 ต้นไม้ : https://jisho.org/search/%E6%9C%A8%20%23kanji
2. 本 หนังสือ : https://jisho.org/search/%E6%9C%AC%20%23kanji
3. 日 วัน : https://jisho.org/search/%E6%97%A5%20%23kanji
4. 明 สว่าง : https://jisho.org/search/%E6%98%8E%20%23kanji
5. 時 ชั่วโมง : https://jisho.org/search/%E6%99%82%20%23kanji

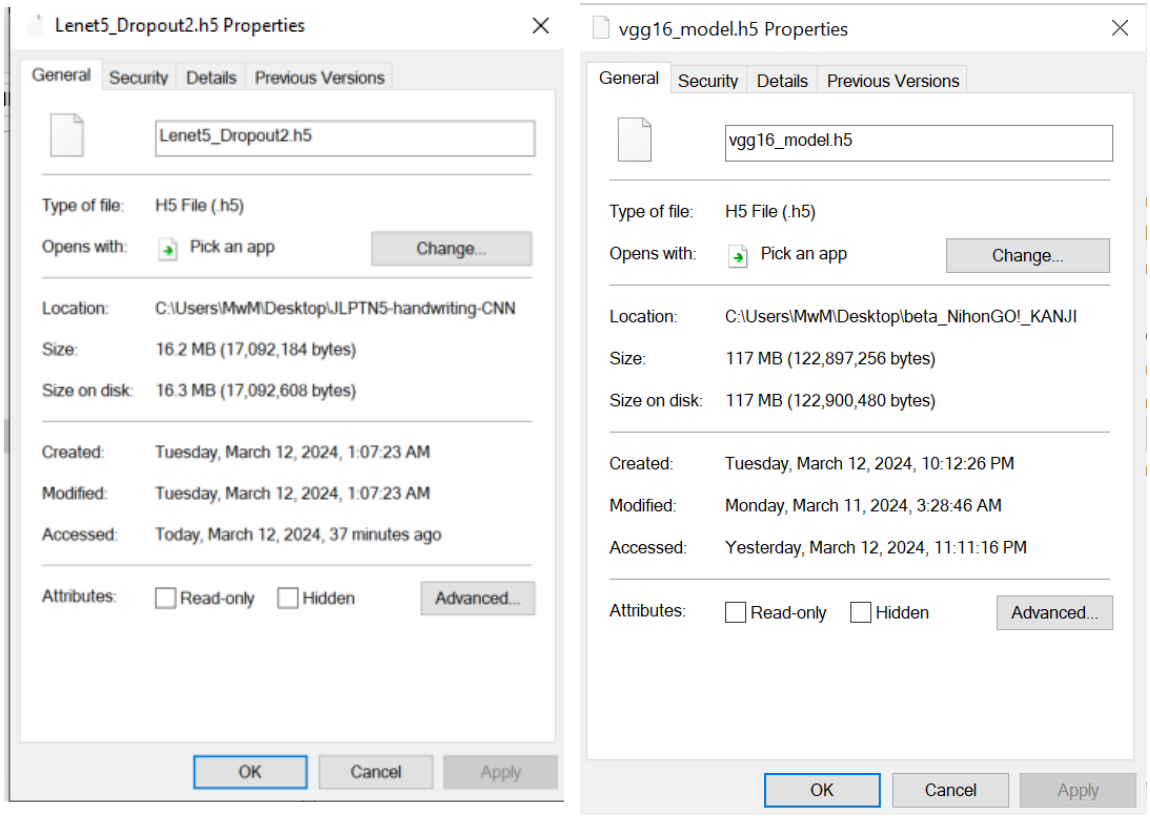
Model	木	本	日	明	時	Total (full of 25)
VGG16	5	5	5	3	3	21 / 25
LeNet-5	5	4	5	4	5	23 / 25

ผลการทดลองจากโมเดล LeNet-5 เทียบกับ VGG16

จากผลการทดลองสรุปได้ว่าจากตัวอย่างตัวอักษรคันจิที่มีลักษณะที่ใกล้เคียงกัน 5 ตัวอักษร โมเดล VGG16 สามารถทำนายได้ 21 ครั้งจากทั้งหมด 25 คิดเป็น 84 % ในขณะเดียวกันจากโมเดล LeNet-5 สามารถทำนายได้ 23 ครั้งจากทั้งหมด 25 คิดเป็น 92 %

4.4) กระบวนการทดสอบโมเดล เกณฑ์ในการเลือกโมเดลที่ดีที่สุด

- จากการทดลองทั้ง 2 โมเดลที่เป็น LeNet-5 และ VGG16 หลังจากเทรนโมเดลพบว่ามีค่า accuracy และค่า loss ที่ใกล้เคียงกันเป็นอย่างมาก ที่ค่าความคาดเคลื่อน +- 0.1 % จึงเลือกโมเดลที่เป็น LeNet-5 เนื่องจากด้วยค่า accuracy ที่เท่ากันของทั้ง 2 models LeNet-5 มีขนาด model ที่เล็กกว่า VGG16 ถึง 7 เท่า เหตุเนื่องมาจากความ deep ของ neural net ซึ่งแปรผลตามความเร็วในการประมวลผล และทรัพยากรตอนใช้งาน



ภาพการเปรียบเทียบขนาดไฟล์โมเดลของ LeNet-5 (ด้านซ้าย) เทียบกับ VGG16 (ด้านขวา)

จากภาพจะเห็นว่าโมเดลของ LeNet-5 มีขนาด 16.2 MB เทียบกับโมเดลของ VGG16 ที่มีขนาด 117 MB เมื่อเทียบกันพบว่าโมเดลมีขนาดต่างกันถึง 7 เท่า

4.5) โมเดลมีความเหมาะสมต่อการแก้ปัญหาหรือไม่ หากไม่ควรมีแนวทางการปรับปรุงอย่างไร

- จากค่า accuracy ค่า loss และจากการทดลองใช้งานจริงหลังจากการ deploy model ผ่าน Huggingface space จากการทดลองกับข้อมูลจากลายมือเขียนจริง ที่ได้จากโมเดลที่เป็น LeNet-5 และ VGG16 สามารถสรุปได้ว่าทั้งโมเดล LeNet-5 และ VGG16 สามารถคัดแยกตัวอักษรคันจิลักษณะเขียนมือได้เป็นอย่างดี กับ มีประสิทธิภาพ อย่างไรก็ตามโมเดลจะทำนายได้ดีก็ต่อเมื่อผู้ใช้งานใส่ข้อมูล input ในลักษณะที่เป็นลายมือเขียนจริง

บทที่ 5
สรุปและอภิปรายผล

5.1) สรุปผลการทดลอง

จากผลการทดลองพบว่า โมเดล LeNet-5 ที่ใช้โครงสร้างของ LeNet-5 โดยกำหนด Dropout เท่ากับ 0.25 สามารถสามารถทำนายผลตัวอักษรคันจิแบบเขียนได้ดีกว่าโมเดล VGG16 โดยกำหนด Dropout เท่ากับ 0.5 ที่มีจำนวน parameters ในการเรียนรู้ที่มากกว่า 15 เท่า แสดงให้เห็นว่าข้อมูลมีความซับซ้อนที่เพียงพอสำหรับ LeNet-5 และจากการทดลองกับข้อมูลจากลายมือเขียนจริง แสดงให้เห็นว่าโมเดลสามารถทำนายผลได้ ซึ่งสอดคล้องกับสมมติฐานที่วางไว้

5.2) อภิปรายผลการทดลอง

ค่าความแม่นยำ (Accuracy) : จากการวิเคราะห์ และวัดผลโมเดลพบว่าค่าความแม่นยำระหว่างโมเดล LeNet-5 และ VGG16 มีค่าความแม่นยำที่ใกล้เคียงกันทั้ง 2 โมเดล ด้วยค่าความคาดเคลื่อนในช่วง +-0.1% จากข้อมูล Test set และ +-12% จากข้อมูลจริงซึ่งแสดงให้เห็นว่าทั้งสองโมเดลมีประสิทธิภาพในการจำแนกข้อมูลที่คล้ายคลึงกัน

ความสามารถในการจำแนก : จากการวิเคราะห์ และวัดผลโมเดลในบทที่ 4 พบว่าในบางกรณีโมเดล LeNet-5 สามารถให้ค่าผลลัพธ์ได้ดีกว่า VGG16 โดยใช้ทรัพยากรที่น้อยกว่า

ขนาดของโมเดล : LeNet-5 มีขนาดเล็กกว่า VGG16 ถึง 7 เท่าเนื่องจากมี depth ของ Neural Network และ จำนวน parameters ที่น้อยกว่า

5.3) ข้อจำกัด และอุปสรรค์

ในกรณีที่ข้อมูล input ที่เข้ามาไม่ตรงกับ labels ที่มีในชุดข้อมูลที่เทรนพบว่า ในบางกรณีโมเดลจะให้ค่าความน่าจะเป็นที่ต่ำกว่า 60 % หรือให้ค่าความน่าจะเป็นไม่ตรงกับ

labels ที่มีอยู่จริง เนื่องจากหลักการทำงานของ Classification คือจะดูองค์ประกอบของภาพรวมทั้งหมดว่ามีความใกล้เคียงกับข้อมูลที่เคยเทรนไว้

ในกรณีที่ข้อมูล input มีขนาดที่ต่างกัน และอยู่ในมุมที่ต่างกันมีผลต่อค่าความแม่นยำในการทำนายของโมเดล เนื่องจาก dataset หลักที่ใช้ในการเทรนมีขนาดที่ตายตัว และตำแหน่งของตัวอักษรของคันจิอยู่ที่ตำแหน่งกลาง ถึงจะมีการทำ Image augmentation ควบคู่ไปด้วยก็ตาม

5.4) ข้อเสนอแนะ และแนวทางในการพัฒนา

- เพื่อให้มั่นใจว่าโมเดลที่เราเทรนนั้นไม่เกิดการ Overfitting เกิดขึ้นควรที่จะมีการทำ K-fold cross validation

- อีกหนึ่งเทคนิคน่าลองใช้คือ Zero-Shot Image Classification หรือ Few-shot Image Classification ที่พัฒนาสำหรับในกรณีทีมีข้อมูลที่ใช้ในการเทรนจำนวนจำกัด

- ในกรณีที่มีผู้เชี่ยวชาญในด้านอักษรศาสตร์ภาษาญี่ปุ่น สามารถพัฒนาต่อยอดโมเดลต่อได้เพื่อใช้ในการเรียนรู้ หรือวิเคราะห์ข้อมูลจากเอกสารต่าง ๆ

บรรณานุกรม

[1] : *Lenet* (2023) *Wikipedia*. Available at: https://en.wikipedia.org/wiki/LeNet (Accessed: 13 March 2024).

[2] : GfG (2023) *VGG-16: CNN model*, *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/vgg-16-cnn-model/ (Accessed: 13 March 2024).

[3] : (No date) *Segmentation of Japanese handwritten characters using peripheral ...https://www.researchgate.net/.../3765056_segmentation_of_japanese_handwritten_char...* Available at: https://www.researchgate.net/publication/3765056_Segmentation_of_Japanese_handwritten_characters_using_peripheral_feature_analysis (Accessed: 12 March 2024).

[4] : (No date a) *Deep learning on AWS*. Available at: https://aws.amazon.com/deep-learning/ (Accessed: 12 March 2024).