



ระบบสกัดข้อมูลจากใบเสร็จอิเล็กทรอนิกส์และใบเสร็จจากเครื่องรับจ่ายเงินอัตโนมัติ (ATM) โดย  
ใช้เทคนิคการรู้จำอักขระด้วยแสง (Optical Character Recognition: OCR)

โดย

นายคุปตาภา วิจารณ์ 6409610596

นางสาวร้อยแก้ว ศิริวัฒน์ 6409610729

โครงการในรายงานวิชาคพ.381 นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

สาขาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยธรรมศาสตร์ พ.ศ. 2568

# สารบัญ

สารบัญ .....	2
บทที่ 1.....	5
บทนำ .....	5
1.1 สาเหตุและที่มาของปัญหา .....	5
1.2 วัตถุประสงค์ของงานวิจัย .....	5
1.3 ขอบเขตงานวิจัย .....	5
บทที่ 2.....	7
ทบทวนวรรณกรรม .....	7
2.1 OpenCV .....	7
2.2 Optical Character Recognition (OCR).....	7
2.3 You Only Look Once (YOLO) version 8 .....	7
2.4 ทฤษฎีที่เกี่ยวข้อง .....	8
2.5 งานวิจัยที่เกี่ยวข้อง .....	15
2.5.1 Extraction of Information from Bill Receipts Using Optical Character Recognition .....	15
2.5.2 Receipt Automatic Reader.....	16
2.5.3 Development of a Process to Enhance the Reimbursement Efficiency with OCR and Ontology for Financial Documents .....	17
2.5.4 Menu Item Extraction from Thai Receipt Images Using Deep Learning and Template-based Information Extraction .....	18
บทที่ 3 .....	19
ระเบียบวิธีการดำเนินงานวิจัย.....	19
3.1 ภาพรวมและสภาพแวดล้อมของการพัฒนาระบบสกัดข้อมูลสำหรับใบเสร็จอิเล็กทรอนิกส์การทำธุรกรรมจากแอปพลิเคชันธนาคาร (Mobile Banking) .....	19
3.1.1 วิธีการพัฒนา.....	19
3.1.1.1 ขั้นตอนการทำ Bank Classification .....	19
3.1.1.2 ขั้นตอนการทำ Optical Character Recognition (OCR) .....	25

3.1.2	โครงสร้างภาพรวมของระบบ .....	28
3.2	ภาพรวมและสภาพแวดล้อมของการพัฒนาระบบสกัดข้อมูลสำหรับใบเสร็จการทำธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ (ATM) .....	29
3.2.1	วิธีการพัฒนา.....	29
3.2.1.1	ขั้นตอนการตรวจจับพื้นที่ใบเสร็จบนภาพ .....	29
1.	กระบวนการเตรียมงานก่อนขั้นตอนการตรวจจับพื้นที่ใบเสร็จบนภาพ .....	29
2.	ขั้นตอนตรวจสอบประเภทของไฟล์รูปภาพ.....	33
3.	ขั้นตอนการตรวจหาใบเสร็จและระบุประเภทธนาคารเจ้าของใบเสร็จ .....	33
4.	ขั้นตอนการเตรียมภาพก่อนขั้นตอน Perspective Transformation .....	35
5.	ขั้นตอนการทำ Perspective Transformation .....	36
3.2.1.2	ขั้นตอนการทำ OCR.....	39
1.	ขั้นตอนการสกัดข้อความจากภาพ .....	39
2.	ขั้นตอนการทำ Text Classification .....	40
3.2.2	โครงสร้างภาพรวมของระบบ .....	47
3.3	การพัฒนาในรูปแบบ Web application.....	48
3.3.1	โครงสร้างภาพรวมของระบบ .....	48
3.4	เครื่องมือที่ใช้ในการพัฒนา .....	50
3.4.1	ภาพนำเข้า.....	50
3.4.2	โปรแกรม Visual Studio Code (VS Code).....	50
3.4.3	Jupyter Notebook.....	51
3.4.4	YOLO V8 .....	51
3.5.1	Use case diagram .....	53
3.5.2	Use case description .....	53
3.5.3	Activity Diagram ของระบบสกัดข้อมูลสำหรับใบเสร็จอิเล็กทรอนิกส์การทำธุรกรรมจากแอปพลิเคชันธนาคาร (Mobile Banking) .....	54
3.5.4	Activity Diagram ของระบบสกัดข้อมูลจากใบเสร็จการทำธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ (ATM) .....	55

บทที่ 4.....	56
การทดลอง .....	56
4.1 เครื่องมือในการทดลอง .....	57
4.2 การทดลองของระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking.....	57
4.2.1 เครื่องมือในการทดลอง.....	57
4.2.2.1 ขั้นตอนในการทดลอง .....	57
4.2.2.2 ผลการทดลอง .....	57
4.2.3 การทดลองเพื่อวัดประสิทธิภาพการสกัดข้อความ.....	58
4.2.3.1 ขั้นตอนในการทดลอง.....	58
4.2.3.2 ผลการทดลอง.....	58
4.3 การทดลองของระบบสกัดข้อความจากใบเสร็จจากตู้ ATM .....	59
4.3.1 เครื่องมือในการทดลอง.....	59
4.3.2.1 ขั้นตอนในการทดลอง .....	59
4.3.2.2 ผลการทดลอง .....	59
4.3.3 การทดลองเพื่อวัดประสิทธิภาพการสกัดข้อความ.....	60
4.3.3.1 ขั้นตอนในการทดลอง.....	60
4.3.3.2 ผลการทดลอง.....	60
บทที่ 5.....	61
สรุป อภิปรายผล และข้อเสนอแนะ .....	61
5.1 สรุปผลการดำเนินงาน .....	61
5.2 อภิปรายผล .....	61
5.3 ข้อจำกัดของระบบ .....	62
5.4 ข้อเสนอแนะ .....	63

# บทที่ 1

## บทนำ

### 1.1 สาเหตุและที่มาของปัญหา

จากผลสำรวจของธนาคารแห่งประเทศไทยในปี 2567 พบว่า คนไทยกว่าร้อยละ 97 เข้าถึงบริการทางการเงิน ไม่ว่าจะเป็นบริการฝาก ถอน หรือโอนเงิน หลังจากการทำธุรกรรมดังกล่าวแล้วแต่จะได้รับหลักฐานหลังการทำธุรกรรมทั้งสิ้น อาทิ ใบเสร็จการฝากหรือถอนเงินจากเครื่องรับจ่ายเงินอัตโนมัติ หรือตู้ ATM และใบเสร็จจากการทำธุรกรรมผ่านแอปพลิเคชัน Mobile Banking ซึ่งใบเสร็จทั้ง 2 ชนิดนี้สามารถนำมาใช้ประโยชน์ต่อในการทำบันทึกรายรับรายจ่าย ซึ่งเป็นประโยชน์ต่อการบริหารจัดการการเงินส่วนบุคคล

การบริหารจัดการการเงินส่วนบุคคลเป็นรากฐานสำคัญของการดำเนินชีวิต อันเปรียบเสมือน “เข็มทิศ” ที่ทำให้เราตัดสินใจได้อย่างชาญฉลาด เพื่อบรรลุเป้าหมายที่ตั้งไว้ อีกทั้งยังเป็นส่วนหนึ่งในการนำเราไปสู่ความมั่งคั่งอย่างใดก็ตาม การจัดการการเงินส่วนบุคคลโดยใช้ใบเสร็จทั้ง 2 ชนิดที่อยู่ในรูปแบบที่ต่างกัน สามารถทำให้เกิดความยุ่งยากในการจัดการได้ เนื่องจากการบันทึกข้อมูลจากใบเสร็จเหล่านี้ยังต้องอาศัยการป้อนข้อมูลด้วยตัวเอง ซึ่งก่อให้เกิดความไม่สะดวก อีกทั้งยังเป็นกระบวนการทำงานที่ซ้ำซ้อนและสิ้นเปลืองเวลา

ในโครงการนี้จะพัฒนาระบบที่สามารถสกัดข้อมูลจากใบเสร็จจากการธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ หรือตู้ ATM และใบเสร็จจากการทำธุรกรรมผ่าน Mobile Banking ผ่านเทคนิคการรู้จำอักขระด้วยแสง (Optical Character Recognition: OCR) เพื่อลดภาระในการป้อนข้อมูลด้วยตนเอง และมีศักยภาพในการนำไปบูรณาการเป็นส่วนหนึ่งของแอปพลิเคชันบันทึกรายรับรายจ่ายผ่านภาพของหลักฐานการทำธุรกรรมในอนาคต

### 1.2 วัตถุประสงค์ของงานวิจัย

- ศึกษาและทำความเข้าใจเทคนิค Digital Image Processing
- เพื่อตรวจจับข้อมูลจากรูปภาพใบเสร็จจากการทำธุรกรรมบน Mobile Banking และจากตู้รับจ่ายเงินอัตโนมัติ (ATM)
- ลดเวลาในการบริหารจัดการการเงินส่วนบุคคล
- เพื่อนำไปเป็นส่วนหนึ่งของการพัฒนาแอปพลิเคชันจัดการการเงินส่วนบุคคลในอนาคต

### 1.3 ขอบเขตงานวิจัย

- รูปของใบเสร็จทั้ง 2 ชนิดเป็นภาพที่ได้จากการถ่ายผ่านกล้องโทรศัพท์มือถือเท่านั้น

- ใบเสร็จมาจาก 4 ธนาคารดังต่อไปนี้เท่านั้น ได้แก่ ธนาคารกรุงเทพ ธนาคารไทยพาณิชย์ ธนาคารกสิกรไทย และธนาคารกรุงไทย

## บทที่ 2

### ทบทวนวรรณกรรม

#### 2.1 OpenCV

OpenCV เป็นไลบรารีโอเพนซอร์สที่ครอบคลุมฟังก์ชันการประมวลผลภาพ และวิทยาศาสตร์คอมพิวเตอร์อย่างกว้างขวาง (a, 2024) ไลบรารีนี้ได้รับการออกแบบมา เพื่อเป็นเครื่องมือพื้นฐานที่จำเป็นสำหรับการแก้ไขปัญหาด้านวิทยาศาสตร์คอมพิวเตอร์ OpenCV รองรับกระบวนการต่างๆ ในการจัดการภาพดิจิทัล เช่น การแก้ไขสี การแปลง การกรอง การเพิ่มขอบ การตรวจจับและระบุวัตถุ และการแบ่งส่วนภาพ (Review on Computer Vision Using OpenCV - ijrpr). ด้วยฟังก์ชันที่ครอบคลุม OpenCV จึงเป็นเครื่องมือสำคัญในการพัฒนาแอปพลิเคชันด้านวิทยาศาสตร์คอมพิวเตอร์ที่ซับซ้อน

#### 2.2 Optical Character Recognition (OCR)

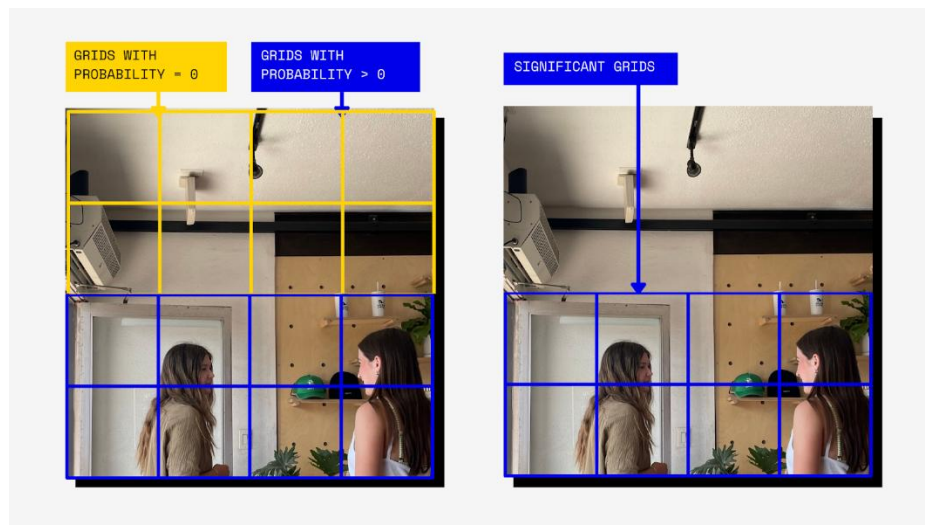
การรู้จำอักขระด้วยแสง (Optical Character Recognition: OCR) เป็นเทคโนโลยีที่สามารถจดจำข้อความในเอกสารหรือภาพดิจิทัลได้โดยอัตโนมัติ. OCR ใช้เทคนิคการประมวลผลภาพและ Machine Learning เพื่อตรวจจับและจดจำตัวอักษร จากนั้นจึงแปลงผลลัพธ์เป็นรูปแบบข้อความดิจิทัล. เทคนิค OCR ที่ใช้ Machine Learning สามารถสกัดข้อความที่พิมพ์หรือเขียนด้วยลายมือจากภาพต่างๆ เช่น ป้าย โปสเตอร์ ฉลากสินค้า รวมถึงเอกสารอย่างบทความ รายงาน แบบฟอร์ม และใบแจ้งหนี้. โดยทั่วไป ข้อความจะถูกสกัดในรูปแบบคำบรรทัดข้อความ ย่อหน้า หรือบล็อกข้อความ ทำให้สามารถเข้าถึงข้อมูลที่สแกนในรูปแบบดิจิทัลได้

#### 2.3 You Only Look Once (YOLO) version 8

YOLOv8 (You Only Look Once version 8) เป็นอัลกอริธึมตรวจจับวัตถุ (Object Detection) ที่ถูกพัฒนาโดย Ultralytics ต่อเนื่องจากตระกูล YOLO รุ่นก่อนหน้า โดยมุ่งเน้นไปที่การเพิ่มความแม่นยำ (Accuracy) และประสิทธิภาพ (Efficiency) ในการตรวจจับวัตถุแบบเรียลไทม์ในภาพและวิดีโอ

YOLOv8 ไม่ได้เป็นเพียงแค่อัลกอริธึมตรวจจับวัตถุเท่านั้น แต่ยังรองรับงานด้าน Computer Vision อื่นๆ ได้แก่ การแบ่งส่วนภาพ (Image Segmentation), การจำแนกประเภทภาพ (Image Classification) และการประมาณท่าทาง (Pose Estimation) ทำให้เป็นเครื่องมือที่หลากหลายและมีประสิทธิภาพสูง

YOLOv8 ทำงานโดยการมองภาพทั้งภาพในครั้งเดียวแล้วแบ่งภาพนั้นออกเป็นตารางเล็ก ๆ ในแต่ละช่องของตาราง โมเดลจะพยายามทำนายว่ามีวัตถุอะไรอยู่บ้าง พร้อมทั้งวาดกรอบสี่เหลี่ยม (Bounding Box) รอบวัตถุนั้น และบอกด้วยว่าเป็นวัตถุนิดไหน เช่น รถยนต์ หรือคน นอกจากนี้ โมเดลยังให้คะแนนความมั่นใจ (Confidence Score) ว่าสิ่งที่ทำนายนั้นถูกต้องมากน้อยแค่ไหน วิธีนี้ทำให้ YOLOv8 สามารถตรวจจับวัตถุได้รวดเร็ว เพราะไม่ต้องไล่ดูภาพทีละส่วนเหมือนวิธีเก่า ๆ แต่จะดูภาพรวมทั้งหมดพร้อมกัน ทำให้เหมาะกับงานที่ต้องการความรวดเร็ว



ภาพประกอบคำอธิบายการทำงานของ YOLOv8

อ้างอิง: <https://www.viam.com/post/guide-yolo-model-real-time-object-detection-with-examples>

## 2.4 ทฤษฎีที่เกี่ยวข้อง

### ○ ภาพไบนารี

เป็นภาพที่แต่ละพิกเซลมีค่าเพียงสองค่าเท่านั้น ซึ่งโดยทั่วไปคือ 0 (สีดำ) และ 1 (สีขาว).

ภาพไบนารีมักถูกสร้างขึ้นจากภาพระดับเทาผ่านกระบวนการแบ่งระดับสี (thresholding)

เพื่อแยกส่วนที่สนใจออกจากพื้นหลัง ซึ่งเป็นประโยชน์อย่างมากในการวิเคราะห์รูปร่างและโครงสร้างของวัตถุ (encord, 2023)

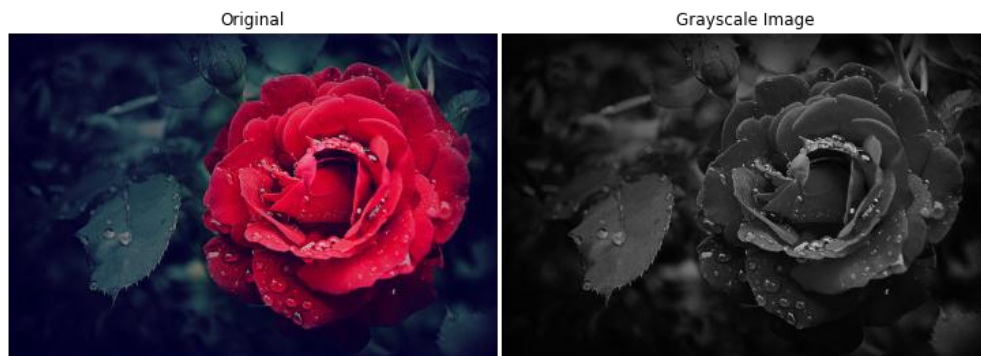




อ้างอิง: <https://pub.towardsai.net/who-needs-photo-editors-massive-image-manipulation-tutorial-in-python-38cb58ad07fe>

#### ○ ภาพระดับเทา

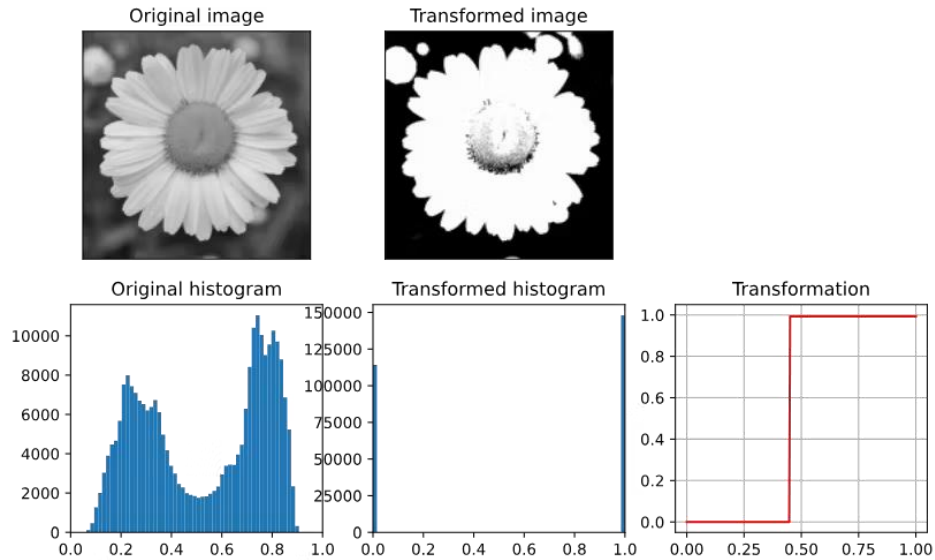
เป็นภาพที่แต่ละพิกเซลมีค่าความเข้มแสงในช่วงหนึ่ง ซึ่งโดยทั่วไปคือ 0 ถึง 255 โดยที่ 0 แทนสีดำสนิท และ 255 แทนสีขาวสนิท. ภาพระดับเทาให้ข้อมูลความเข้มแสงที่ละเอียดกว่าภาพไบนารี และมักเป็นขั้นตอนกลางในการประมวลผลภาพก่อนที่จะแปลงเป็นภาพไบนารี



อ้างอิง: <https://pub.towardsai.net/who-needs-photo-editors-massive-image-manipulation-tutorial-in-python-38cb58ad07fe>

#### ○ Thresholding

เป็นเทคนิคที่ใช้ในการแปลงภาพระดับเทาให้เป็นภาพไบนารี โดยการกำหนดค่าขีดแบ่ง (threshold value). พิกเซลที่มีค่าความเข้มแสงสูงกว่าขีดแบ่งจะถูกกำหนดให้เป็นสีขาว (หรือค่า 1) ในขณะที่พิกเซลที่มีค่าความเข้มแสงต่ำกว่าขีดแบ่งจะถูกกำหนดให้เป็นสีดำ (หรือค่า 0) การแบ่งระดับสีช่วยลดความซับซ้อนของภาพและทำให้ง่ายต่อการแยกวัตถุที่สนใจออกจากพื้นหลัง



อ้างอิง: <https://encord.com/blog/image-thresholding-image-processing/>

#### ○ Adaptive Thresholding

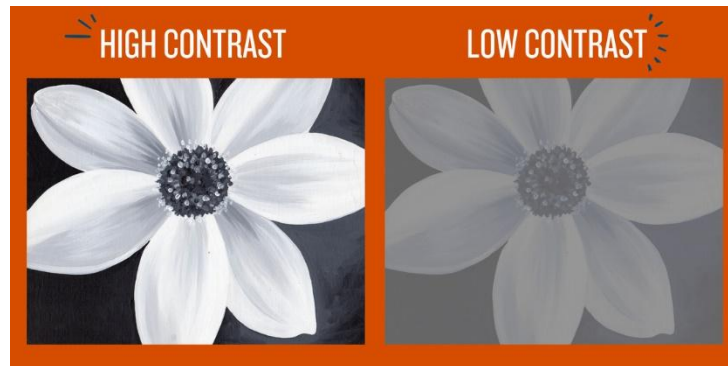
เป็นเทคนิคที่ลดจุดอ่อนของการทำ Thresholding แบบปกติที่ใช้ค่า Threshold เดียวทั้งภาพ (Global Thesholding) เพราะฉะนั้นจึงไม่เหมาะกับภาพที่แสงในภาพไม่คงที่ Adaptive Thresholding จะเข้ามาลดจุดอ่อนตรงนี้ โดยเทคนิคนี้ใช้การหาค่า Local Threshold ซึ่งคือค่า Threshold ของบริเวณหนึ่ง ๆ ถ้าเปลี่ยนบริเวณค่า Threshold ก็จะไม่เปลี่ยน ทำให้สามารถจัดการกับปัญหาแสงที่ไม่เท่ากันทั่วทั้งภาพได้ [https://expert-programming-tutor.com/tutorial/article/KE003582\\_Image\\_Processing\\_with\\_OpenCV\\_-\\_Adaptive\\_Thresholding.php](https://expert-programming-tutor.com/tutorial/article/KE003582_Image_Processing_with_OpenCV_-_Adaptive_Thresholding.php)



อ้างอิง: [https://en.wikipedia.org/wiki/Thresholding\\_%28image\\_processing%29](https://en.wikipedia.org/wiki/Thresholding_%28image_processing%29)

## ○ Contrast

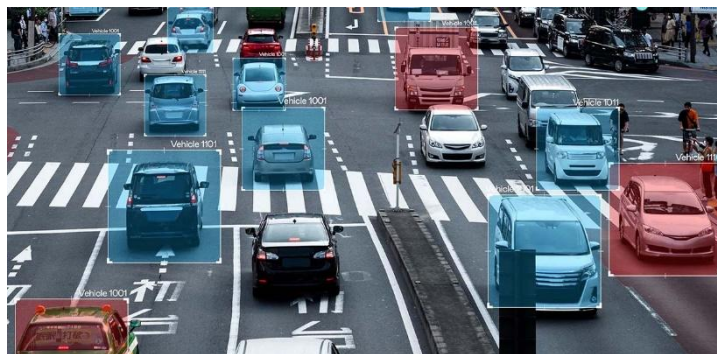
Contrast คือ ความแตกต่างระหว่างส่วนประกอบบนภาพ ภาพที่ Contrast สูงคือภาพที่ขอบของวัตถุตัดกันชัดเจน ส่วนภาพที่ Contrast ต่ำ คือ ภาพที่ขอบของวัตถุตัดกันไม่ชัดเจน



อ้างอิง: <https://www.thesketchingpad.com/blog/why-contrast-is-the-key-to-visually-appealing-art>

## ○ Bounding box

คือสี่เหลี่ยมผืนผ้าที่เล็กที่สุดที่สามารถล้อมรอบวัตถุที่ตรวจพบในภาพได้. Bounding box ถูกกำหนดโดยพิกัดของมุมซ้ายบนและความกว้างและความสูงของสี่เหลี่ยม ซึ่งเป็นประโยชน์ในการระบุตำแหน่งและขนาดของวัตถุในภาพ



อ้างอิง: <https://www.analytics.ai/blog/bounding-box-annotation-importance-types-tips/>

## ○ Region of interest (ROI)

คือส่วนใดส่วนหนึ่งของภาพที่ผู้ใช้หรือระบบต้องการเน้นหรือวิเคราะห์เป็นพิเศษ. การกำหนด ROI ช่วยให้สามารถจำกัดการประมวลผลไปเฉพาะส่วนที่เกี่ยวข้องเท่านั้น ซึ่งช่วยประหยัดเวลาและทรัพยากรในการคำนวณ

## ○ Template Matching

Template matching คือกระบวนการค้นหาลักษณะในภาพขนาดใหญ่ (source image) ที่ตรงกับภาพขนาดเล็ก (template image) โดยการเลื่อน template image ไปบน source image ที่ละพิกเซล และคำนวณค่าความคล้ายคลึงกัน (similarity) หรือความแตกต่าง (difference) ระหว่าง template กับส่วนของ source image ที่ถูกทับอยู่ ณ ตำแหน่งนั้นๆ

OpenCV มีวิธีการเปรียบเทียบ template matching อยู่หลายวิธี แต่ละวิธีมีสมการในการคำนวณค่า  $R(x,y)$  ที่แตกต่างกัน โดยที่  $(x,y)$  คือพิกัดมุมซ้ายบนของบริเวณที่เปรียบเทียบใน source image,  $I$  คือ source image,  $T$  คือ template image และ  $M$  คือ mask (ถ้ามี)

วิธี TM\_CCOEFF (Correlation Coefficient) วัดค่าสหสัมพันธ์ระหว่าง template และ source image โดยมีการลบค่าเฉลี่ยออกก่อน ค่ายิ่งมากแสดงว่ายิ่งคล้ายกันมาก

$$R(x, y) = \sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))$$

where

$$T'(x', y') = T(x', y') - 1/(w \cdot h) \cdot \sum_{x'', y''} T(x'', y'')$$

$$I'(x + x', y + y') = I(x + x', y + y') - 1/(w \cdot h) \cdot \sum_{x'', y''} I(x + x'', y + y'')$$

with mask:

$$T'(x', y') = M(x', y') \cdot \left( T(x', y') - \frac{1}{\sum_{x'', y''} M(x'', y'')} \cdot \sum_{x'', y''} (T(x'', y'') \cdot M(x'', y'')) \right)$$

$$I'(x + x', y + y') = M(x', y') \cdot \left( I(x + x', y + y') - \frac{1}{\sum_{x'', y''} M(x'', y'')} \cdot \sum_{x'', y''} (I(x + x'', y + y'') \cdot M(x'', y'')) \right)$$

อ้างอิง: [https://docs.opencv.org/4.x/df/dfb/group\\_\\_imgproc\\_\\_object.html](https://docs.opencv.org/4.x/df/dfb/group__imgproc__object.html)

โดยในโปรเจกต์นี้เลือกใช้วิธี TM\_CCOEFF\_NORMED (Normalized Correlation Coefficient)

$$R(x, y) = \frac{\sum_{x', y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x', y'} T'^2(x', y') \cdot \sum_{x', y'} I'^2(x + x', y + y')}}^2$$

อ้างอิง: [https://docs.opencv.org/4.x/df/dfb/group\\_\\_imgproc\\_\\_object.html](https://docs.opencv.org/4.x/df/dfb/group__imgproc__object.html)

เหมือน TM\_CCOEFF แต่มีการทำ Normalization เพื่อให้ค่าอยู่ระหว่าง -1 ถึง 1 ค่ายิ่งมาก (เข้าใกล้ 1) แสดงว่ายิ่งคล้ายกันมาก ค่า 0 แสดงว่าไม่มีความสัมพันธ์ และค่าติดลบแสดงว่ามีความสัมพันธ์แบบผกผัน วิธีนี้มักจะให้ผลลัพธ์ที่แม่นยำที่สุดและทนทานต่อการเปลี่ยนแปลงของแสงและความสว่าง

$I$  source image (ภาพต้นฉบับ)

$T$  template image (ภาพแม่แบบ)

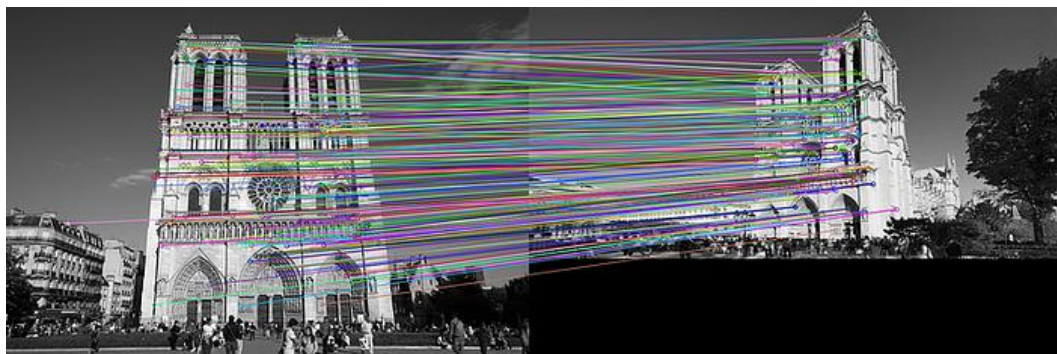
$R$	ผลลัพธ์ของการเปรียบเทียบ (result matrix)
$M$	mask (ถ้ามี) เป็นภาพขาวดำที่ระบุว่าพิกเซลใดใน template ที่จะนำมาใช้ในการคำนวณ
$(x, y)$	พิกัดมุมซ้ายบนของบริเวณที่กำลังเปรียบเทียบใน source image
$(x', y')$	พิกัดภายใน template image
$\Sigma_{x', y'}$	ผลรวมของค่าต่างๆ ของทุกพิกเซลใน template
$w, h$	ความกว้างและความสูงของ template



อ้างอิง: [https://docs.opencv.org/4.x/d4/dc6/tutorial\\_py\\_template\\_matching.html](https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html)

#### ○ FLANN (Fast Library for Approximate Nearest Neighbors) Matching

เป็นไลบรารีสำหรับการค้นหา Neighborhood ที่ใกล้เคียงที่สุดโดยประมาณในชุดข้อมูลขนาดใหญ่และในพื้นที่หลายมิติ ออกแบบมาเพื่อเลือกอัลกอริทึมและพารามิเตอร์ที่เหมาะสมที่สุดโดยอัตโนมัติ ขึ้นอยู่กับชุดข้อมูลนั้นๆ FLANN Based Matcher ใน OpenCV เป็นเครื่องมือที่ใช้ในการจับคู่หรือค้นหาคุณลักษณะ (features) ของภาพหนึ่งกับอีกภาพหนึ่ง ซึ่งมักจะทำงานได้เร็วกว่า BruteForceMatcher สำหรับชุดข้อมูลที่หลากหลายและมีขนาดใหญ่ โดยทั่วไปแล้ว



อ้างอิง: <https://medium.com/@lucasmassucci/exploring-correlations-in-images-with-sift-and-flann-an-efficient-approach-to-feature-matching-1fdb33697f5e>

การทำงานของ FLANN Matching สามารถสรุปได้ดังนี้:

#### 1. Indexing

FLANN จะสร้างโครงสร้างข้อมูลที่เรียกว่า index จากชุดของคุณลักษณะ (descriptors) ของภาพฝึกฝน (train image) หรือชุดข้อมูลอ้างอิง มีหลายอัลกอริทึมในการสร้าง index ที่ FLANN รองรับ เช่น KD-Trees, KMeans njia uliyotumia kuandika ujumbe huu (KMeans clustering) และ LSH (Locality Sensitive Hashing) การเลือกอัลกอริทึมที่เหมาะสมขึ้นอยู่กับลักษณะของข้อมูลและข้อกำหนดด้านประสิทธิภาพ การ

#### 2. Nearest Neighbor Search

เมื่อมีคุณลักษณะจาก source image เข้ามา FLANN จะใช้ index ที่สร้างไว้เพื่อค้นหาคุณลักษณะที่ "คล้ายคลึง" หรือ "ใกล้เคียง" ที่สุดในชุดข้อมูลอ้างอิง

"ความใกล้เคียง" นี้วัดจากระยะห่างระหว่างคุณลักษณะ โดยทั่วไปใช้ระยะทางแบบ Euclidean (L2-norm) สำหรับคุณลักษณะเช่น SIFT หรือ SURF หรือใช้ Hamming distance สำหรับคุณลักษณะแบบไบนารี เช่น ORB หรือ BRISK

#### 3. Matching

FLANN Based Matcher จะทำการจับคู่คุณลักษณะจากภาพสอบถามกับคุณลักษณะในภาพฝึกฝนโดยใช้วิธีการค้นหาเพื่อนบ้านที่ใกล้เคียงที่สุด

โดยทั่วไปจะใช้ knnMatch() ซึ่งจะค้นหา k เพื่อนบ้านที่ใกล้เคียงที่สุดสำหรับแต่ละคุณลักษณะของภาพสอบถาม (k มักจะเป็น 2)

#### 4. Filtering Matches

เพื่อให้ได้ผลการจับคู่ที่ดีและลดจำนวนการจับคู่ที่ผิดพลาด (false matches) มักจะมีการใช้เทคนิคการกรองผลลัพธ์

Ratio Test ของ Lowe: เป็นเทคนิคที่นิยมใช้ โดยจะคำนวณอัตราส่วนของระยะทางของเพื่อนบ้านที่ใกล้เคียงที่สุดอันดับแรกกับเพื่อนบ้านที่ใกล้เคียงที่สุดอันดับสอง หากอัตราส่วนนี้น้อยกว่าค่าเกณฑ์ที่กำหนด (เช่น 0.7 หรือ 0.75) การจับคู่นั้นจะถือว่าเป็น "การจับคู่ที่ดี" แนวคิดคือถ้าการจับคู่มีความชัดเจน ระยะห่างของ Nearest Neighborhood อันดับแรกจะน้อยกว่าระยะห่างของ Nearest Neighborhood อันดับสองอย่างมีนัยสำคัญ

Cross Check Test: การจับคู่ (fa, fb) จะถือว่าดีก็ต่อเมื่อ fb เป็น Nearest Neighborhood สำหรับ fa ในภาพที่สอง และ fa ก็เป็นเพื่อนบ้านที่ดีที่สุดสำหรับ fb ในภาพแรกด้วย

Geometric Test: กำจัดการจับคู่ที่ไม่สอดคล้องกับแบบจำลองทางเรขาคณิต เช่น RANSAC หรือ robust homography สำหรับวัตถุระนาบ

## 2.5 งานวิจัยที่เกี่ยวข้อง

### 2.5.1 Extraction of Information from Bill Receipts Using Optical Character Recognition

งานวิจัยนี้ถูกนำเสนอโดย Vedant Kumar, Pratyush Kaware, Pradhuman Singh, Reena Sonkusare, และ Siddhant Kumar ในปี 2020 งานวิจัยฉบับนี้นำเสนอการพัฒนาแอปพลิเคชันสำหรับสกัดข้อมูลจากใบเสร็จรับเงิน โดยใช้เทคโนโลยีการรู้จำอักขระด้วยแสง (Optical Character Recognition: OCR) ระบบที่พัฒนาขึ้นผสมผสานการประมวลผลภาพด้วย OpenCV ร่วมกับ Tesseract OCR เพื่อทำการสกัดข้อความจากภาพใบเสร็จ โดยแอปพลิเคชันได้รับการออกแบบให้สามารถทำงานในรูปแบบออฟไลน์บนระบบปฏิบัติการแอนดรอยด์ เพื่อเพิ่มความสะดวกในการแปลงใบเสร็จจากรูปแบบกระดาษเป็นข้อมูลดิจิทัลอย่างมีประสิทธิภาพ

ในกระบวนการประมวลผลภาพ ระบบได้นำเทคนิค Digital Image Processing มาใช้เพื่อจัดการกับปัญหาทั่วไป เช่น ลายน้ำ เงาม และตราประทับ โดยอาศัยกระบวนการแปลงภาพเป็นภาพระดับสีเทา (grayscale), การ thresholding และการปรับปรุงโครงสร้างภาพด้วยเทคนิค morphological transformations สำหรับใบเสร็จที่มีความยาวเกิน 26 เซนติเมตร ระบบจะทำการแบ่งภาพออกเป็นหลายส่วนที่มีการซ้อนทับกันเล็กน้อย เพื่อให้สามารถสกัดข้อความได้อย่างถูกต้องและครบถ้วน

การสกัดข้อความในระบบนี้ใช้ Tesseract OCR ซึ่งใช้โครงข่ายประสาทเทียมแบบ Convolutional Neural Network (CNN) และ Long Short-Term Memory (LSTM) ในการวิเคราะห์และดึงข้อมูลตัวอักษรจากภาพ นอกจากนี้ ยังมีการประยุกต์ใช้ Regular Expressions เพื่อค้นหาข้อมูลสำคัญ เช่น วันที่และจำนวนเงิน จากข้อความที่ถูกสกัดออกมา

ประสิทธิภาพของระบบได้รับการประเมินผ่านตัวชี้วัดความแม่นยำ (Accuracy) และอัตราส่วนสัญญาณต่อสัญญาณรบกวนสูงสุด (Peak Signal-to-Noise Ratio: PSNR) ซึ่งผลการทดลองแสดงให้เห็นว่าสามารถจัดการกับปัญหาลายน้ำ เงาม และตราประทับได้อย่างมีประสิทธิภาพ ระบบที่พัฒนาขึ้นสามารถลดเวลาในการจัดการเอกสารทางการเงินในรูปแบบกระดาษ และช่วยให้ผู้ใช้งานสามารถเข้าถึงและจัดระเบียบข้อมูลธุรกรรมได้สะดวกและรวดเร็วยิ่งขึ้น

นอกจากนี้ งานวิจัยยังชี้ให้เห็นถึงความท้าทายในการสกัดข้อมูลจากใบเสร็จเก่า เนื่องจากข้อความมักมีความซีดจาง ซึ่งอาจแก้ไขได้ในอนาคตด้วยการประยุกต์ใช้เทคโนโลยี Augmented Reality (AR) และ Virtual Reality (VR) เพื่อช่วยในการฟื้นฟูข้อความที่เลือนราง





Raw Image

Processed Image

ภาพที่ .. เปรียบเทียบภาพต้นฉบับและภาพที่ถูกปรับปรุงแล้ว

## 2.5.2 Receipt Automatic Reader

งานวิจัยนี้ถูกนำเสนอโดย Olga Maslova, Louis Klein, Damien Dabernat, Alexandre Benoit, และ Patrick Lambert ในปี 2019 โดยนำเสนอระบบอ่านข้อมูลจากใบเสร็จโดยอัตโนมัติที่นำเสนอในงานวิจัยนี้ เป็นระบบที่ผสมผสานการทำงานของโครงข่ายประสาทเทียมเชิงลึก (Deep Neural Networks) เข้ากับเทคนิคการประมวลผลภาพแบบดั้งเดิม เพื่อเพิ่มประสิทธิภาพในการตรวจจับและดึงข้อมูลจากใบเสร็จในรูปแบบอัตโนมัติ

กระบวนการเริ่มต้นด้วยขั้นตอนการระบุตำแหน่งและแยกส่วนใบเสร็จ (Receipt Localization & Segmentation) โดยใช้ Mask Region-Based Convolutional Neural Network (Mask-RCNN) ที่มีโครงข่ายหลักเป็น ResNet-101 พร้อมการปรับใช้ฟังก์ชัน Weighted Dice Loss และเทคนิค Sliding Kernel Averaging (ขนาดเคอร์เนล  $k = 11$ ) เพื่อเพิ่มความแม่นยำในการจำแนกขอบเขตของใบเสร็จจากพื้นหลัง

หลังจากนั้น ระบบจะดำเนินการปรับแก้รูปร่างเชิงเรขาคณิตของภาพใบเสร็จ (Geometric Transformation) โดยอาศัย Homography Transformation ร่วมกับโครงข่าย VGG-16 เพื่อจัดแนวทิศทางของใบเสร็จให้อยู่ในมุมที่เหมาะสม เช่น  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$  และ  $270^\circ$

ในขั้นตอนการตรวจจับบล็อกข้อความ (Text Block Detection) ระบบจะทำการปรับขนาดภาพให้มีความกว้าง 512 pixel แล้วประมวลผลด้วย Sobel Filters, เทคนิค Otsu Thresholding และเทคนิค Morphological Closing (ขนาด kernel  $8 \times 3$ ) เพื่อแยกส่วนของข้อความออกจากพื้นหลังอย่างมีประสิทธิภาพ



สำหรับการรู้จำอักขระ (Optical Character Recognition: OCR) ระบบจะประมวลผลในระดับบล็อกข้อความ แทนการประมวลผลแบบทั้งใบเสร็จหรือระดับอักขระเดี่ยว โดยใช้ Google Vision API ซึ่งมีความแม่นยำสูงในการรู้จำตัวอักษรจากภาพ

ผลการทดสอบแสดงให้เห็นว่าระบบสามารถจัดการกับใบเสร็จที่มีคุณภาพต่ำ เช่น ใบเสร็จที่ยับ ย้อนแสง หรือมีการเอียง ได้ดีกว่าระบบ OCR ทั่วไป โดยการประเมินประสิทธิภาพเน้นไปที่ระดับความแม่นยำในการรู้จำข้อความ ซึ่งถือเป็นองค์ประกอบสำคัญสำหรับการนำข้อมูลไปใช้ในกระบวนการวิเคราะห์เชิงความหมาย (Semantic Analysis) ในลำดับถัดไป



Bad quality receipts, with various damages and flaws

ภาพที่ .. ตัวอย่างภาพใบเสร็จคุณภาพต่ำที่งานนี้ใช้

## 2.5.3 Development of a Process to Enhance the Reimbursement Efficiency with OCR and Ontology for Financial Documents

งานวิจัยนี้ถูกนำเสนอโดย Patiyuth Pramkeaw, Narumol Chumuang, Mahasak Ketcham, Worawut Yimyam, และ Thittaporn Ganokratanaa ในปี 2022 งานวิจัยนำเสนอระบบที่ถูกพัฒนาขึ้นเพื่อเพิ่มประสิทธิภาพในการตรวจสอบเอกสารทางการเงินที่ใช้สำหรับการเบิกจ่าย โดยปกติแล้วกระบวนการนี้มักต้องใช้เวลาอันยาวนานกว่า 1 สัปดาห์ เนื่องจากต้องพึ่งพาการส่งเอกสารและใบเสร็จแบบ manual ให้กับฝ่ายบัญชี ซึ่งเสี่ยงต่อความผิดพลาดที่เกิดจากมนุษย์ งานวิจัยนี้จึงเสนอระบบที่ช่วยให้ผู้ใช้งานสามารถถ่ายภาพใบเสร็จผ่านสมาร์ทโฟน จากนั้นระบบจะนำภาพไปประมวลผลด้วย OCR เพื่อดึงข้อมูลตัวอักษรออกจากภาพ พร้อมใช้เทคนิค pre-processing เช่น การแปลงภาพเป็น grayscale และการปรับ contrast เพื่อเพิ่มความแม่นยำในการรู้จำตัวอักษร หลังจากนั้นข้อมูลที่ถูกดึงออกมาจะถูกตรวจสอบความถูกต้องโดยใช้ Ontology ซึ่งจะช่วย map ค่าที่รู้จำได้เข้ากับประเภทข้อมูลที่คาดหวัง เช่น ราคา หรือวันที่ และมีการปรับปรุงความถูกต้องเพิ่มเติมด้วยอัลกอริทึม Levenshtein Distance รวมถึงการตรวจสอบทางคณิตศาสตร์ เช่น การคำนวณยอดรวม

ผลการทดลองพบว่าระบบสามารถลดเวลาในการประมวลผลเอกสารเบิกจ่ายได้มากกว่า 50% โดยมีความแม่นยำอยู่ที่ประมาณ 90–95% สำหรับใบเสร็จคุณภาพดี และ 70–80% สำหรับใบเสร็จคุณภาพต่ำ งานวิจัยสรุปว่าการผสานเทคโนโลยี OCR ร่วมกับ Ontology ช่วยให้กระบวนการเบิกจ่ายมีความรวดเร็วและแม่นยำมากขึ้น แม้ว่าจะยังมีความท้าทายอยู่บ้างในกรณีของใบเสร็จที่มีรูปแบบไม่มาตรฐานหรือมีคุณภาพของภาพต่ำ สำหรับการพัฒนาต่อไปในอนาคต อาจขยายการรองรับประเภทเอกสารเพิ่มเติม และพัฒนาระบบการจัดการผู้ใช้งานให้ตอบโจทย์การใช้งานในระดับองค์กรได้อย่างมีประสิทธิภาพยิ่งขึ้น

#### 2.5.4 Menu Item Extraction from Thai Receipt Images Using Deep Learning and Template-based Information Extraction

งานวิจัยนี้นำเสนอโดย Chompunut A-Sawaareekun และ Rajalida Lipikorn ในปี 2025 งานวิจัยนี้นำเสนอระบบที่ออกแบบมาเพื่อดึงข้อมูลรายการอาหารและข้อมูลการชำระเงินจากใบเสร็จร้านอาหารไทย โดยใช้กระบวนการ 4 ขั้นตอนที่ทำงานระหว่าง deep learning และเทคนิคแบบ rule-based ขั้นตอนแรกเริ่มจากการ preprocessing ภาพใบเสร็จ โดยแปลงเป็น grayscale และใช้ edge detection ร่วมกับ morphological operations เพื่อเน้นบริเวณที่เป็นข้อความและลดสัญญาณรบกวนพื้นหลัง ขั้นตอนที่สองคือการตรวจจับรูปแบบ layout ของใบเสร็จ ซึ่งจำแนกได้เป็น 3 ประเภท ได้แก่ two-column, three-column แบบใช้เลขอารบิก (3-col-arp) และ three-column แบบใช้เลขไทยร่วมกับอารบิก (3-col-nap) โดยจากการทดสอบโมเดลหลายแบบ ได้แก่ Faster R-CNN, YOLOv5 และ YOLOv8 พบว่า YOLOv8 ให้ผลลัพธ์ที่ดีที่สุด ด้วยค่า mean average precision (mAP@0.5:0.95) เท่ากับ 0.835

เมื่อระบบสามารถตรวจจับ layout ได้แล้ว ขั้นตอนที่สามคือการนำส่วนที่สนใจ (เช่น รายการอาหารและสรุปยอดเงิน) ไปผ่านกระบวนการ OCR ด้วย Tesseract เพื่อแปลงภาพเป็นข้อความ ซึ่งก่อนหน้านี้จะมีการเพิ่มคุณภาพภาพให้เหมาะสมกับการอ่านของ OCR โดยเฉพาะในกรณีที่คุณภาพภาพต่ำ ขั้นตอนสุดท้ายคือการประมวลผลข้อความที่ได้จาก OCR ด้วยกฎที่สอดคล้องกับ layout ที่ตรวจพบ โดยใช้ regular expressions และการเปรียบเทียบแบบ fuzzy matching เพื่อดึงข้อมูลให้อยู่ในรูปแบบที่มีโครงสร้าง เช่น ชื่อเมนู ปริมาณ ราคาต่อหน่วย และราคารวม รวมถึงข้อมูลการชำระเงิน เช่น ยอดก่อนภาษี VAT และยอดรวมสุทธิ

จากการประเมินผลบนชุดข้อมูลใบเสร็จ 30 ใบ ระบบสามารถดึงข้อมูลเมนูและราคาได้ด้วย precision 87.37% recall 89.25% และ F1-score 88.30% ส่วนความแม่นยำในการดึงข้อมูลการชำระเงินอยู่ที่ 70.45% แม้ว่าระบบจะให้ผลลัพธ์ที่น่าพอใจ แต่ก็ยังเผชิญกับข้อจำกัด เช่น การจัดการข้อความที่มีทั้งภาษาไทยและอังกฤษในบรรทัดเดียวกัน รูปแบบใบเสร็จที่หลากหลาย และข้อผิดพลาดจาก OCR เมื่อภาพมีคุณภาพต่ำ ผู้เขียนเสนอให้มีการพัฒนาต่อในด้าน preprocessing การปรับปรุงความแม่นยำของ OCR และเพิ่มความยืดหยุ่นให้กับขั้นตอนการดึงข้อมูลเพื่อรองรับรูปแบบใบเสร็จที่หลากหลายมากขึ้นในอนาคต

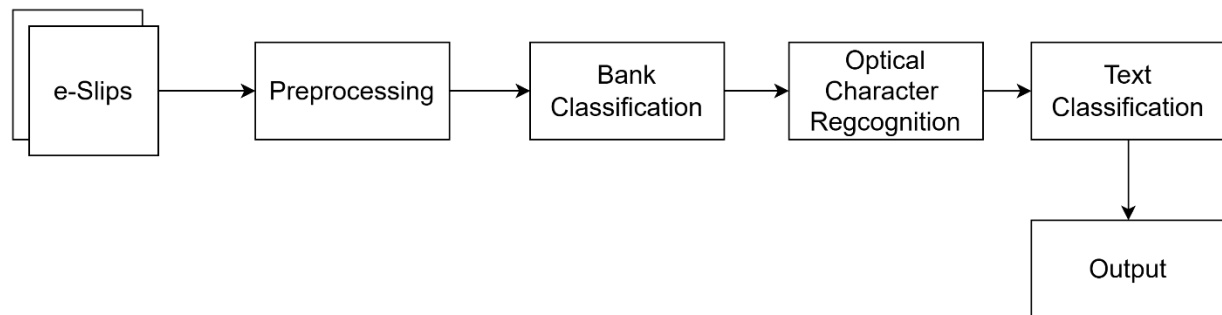
## บทที่ 3

### ระเบียบวิธีการดำเนินงานวิจัย

3.1 ภาพรวมและสภาพแวดล้อมของการพัฒนาระบบสกัดข้อมูลสำหรับใบเสร็จอิเล็กทรอนิกส์การทำธุรกรรมจากแอปพลิเคชันธนาคาร (Mobile Banking)

#### 3.1.1 วิธีการพัฒนา

การพัฒนาแบ่งเป็น 2 ขั้นตอนใหญ่ๆด้วยกันของขั้นตอนคือ 1. การทำ Bank Classification และ 2. การทำ OCR โดยมีขั้นตอนย่อยดังภาพ



ภาพรวมการพัฒนาระบบในส่วนใบเสร็จการทำธุรกรรมอิเล็กทรอนิกส์

##### 3.1.1.1 ขั้นตอนการทำ Bank Classification

###### 1. Data Preparation

เริ่มจากการเตรียม logo ของธนาคารต่างๆ เพื่อนำมาทำ template matching และ FLANN matching โดยเลือกรูปภาพมาดังนี้

###### 1.2 ธนาคารกรุงเทพ



logo ใบเสร็จอิเล็กทรอนิกส์ธนาคารกรุงเทพ

###### 1.3 ธนาคารกสิกร



logo ใบเสร็จอิเล็กทรอนิกส์ธนาคารกสิกร

### 1.3 ธนาคารกรุงไทย



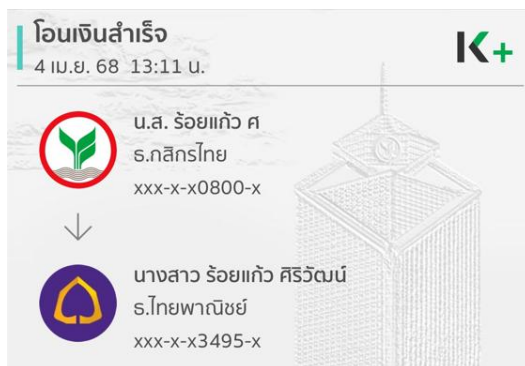
logo ใบเสร็จอิเล็กทรอนิกส์ธนาคารกรุงไทย

### 1.3 ธนาคารไทยพาณิชย์



logo ใบเสร็จอิเล็กทรอนิกส์ธนาคารไทยพาณิชย์

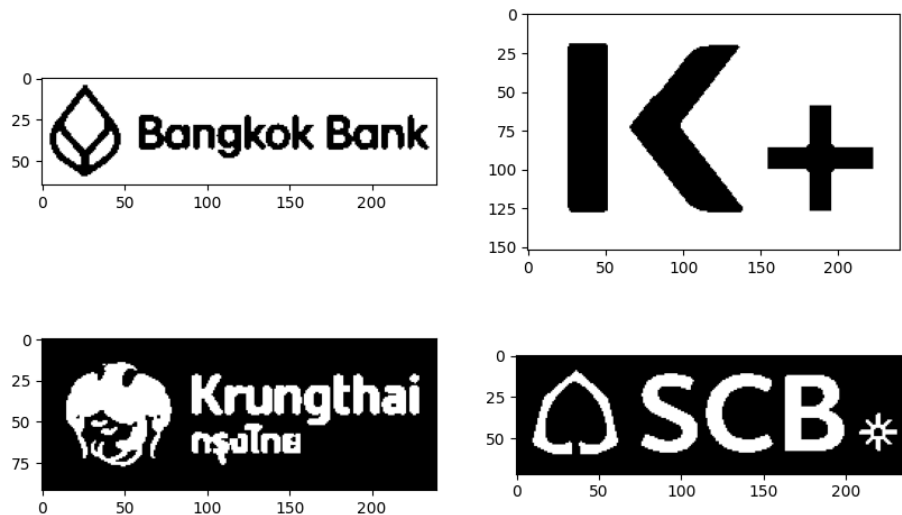
โดยการเลือก logo เราเลือกนำ logo แบบที่มีตัวอักษรอยู่ด้วยเพื่อเพิ่มความแม่นยำในการทำ matching มากขึ้น เนื่องจากถ้าตัด logo มาอย่างเดียวมีโอกาสทำให้ match ผิดพลาดได้ตัวอย่าง เช่น e-slip ของธนาคารกสิกรที่โอนไปยังธนาคารไทยพาณิชย์ จะมี logo ของทั้งสองธนาคารอยู่ในนั้นทำให้มีโอกาสที่โปรแกรมจะตรวจจับธนาคารไทยพาณิชย์แทนได้



ตัวอย่างใบเสร็จธนาคารกสิกรที่โอนไปที่ธนาคารไทยพาณิชย์

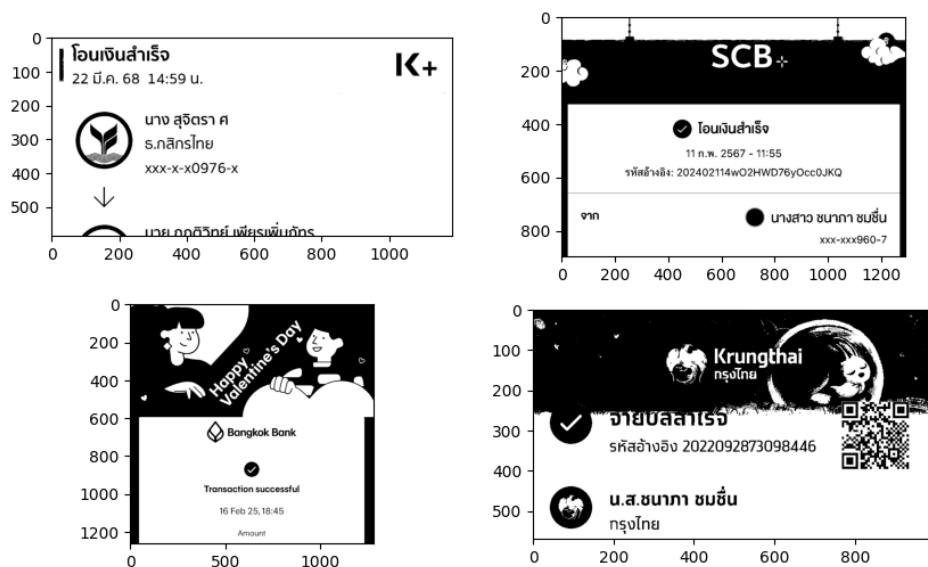
## 2. Preprocessing

ก่อนนำไปทำ Bank Classification จะทำการ Preprocessing โดยการทำให้รูปภาพเป็น gray scale, gaussian blur และทำ Otsu thresholding เพื่อทำให้รูปภาพเป็น binary เพื่อลดทอนความซับซ้อนของภาพ ก่อนนำไปประมวลผล ทั้งในใบเสร็จและ logo ธนาคาร



Logo ธนาคารทั้งหมดหลัง preprocess

ตัดครึ่งล่างของภาพใบเสร็จออกเนื่องจากขั้นตอนนี้เราสนใจแค่ logo ของธนาคารนั้นๆ ซึ่งมักจะอยู่ด้านบนของใบเสร็จ และ ลดข้อผิดพลาดลง



ใบเสร็จอิเล็กทรอนิกส์ของธนาคารทั้งหมดหลัง preprocess

### 3. Bank annotation

#### 3.1 Template Matching

ในการทำ Template Matching จะนำรูปภาพของใบเสร็จจ้อเล็กทรอนิกส์ (source image) และ logo (template) มาเทียบกัน และส่งผลลัพธ์ที่ดีที่สุดออกมา โดยใช้ OpenCV matchTemplate

Template Matching มีข้อจำกัดคือรูป Template และ object ภายใน Source Image ต้องมีขนาดเท่ากันถึงจะสามารถตรวจจับได้ จึงได้ทำ function multi\_scale\_template\_matching() ขึ้นมา

```
def multi_scale_template_matching(image, template, scales=[0.5, 0.75, 1.0, 1.25, 1.5]):
```

Function multi\_scale\_template\_matching

```
for scale in scales:
    # Resize template
    width = int(template.shape[1] * scale)
    height = int(template.shape[0] * scale)
    resized_template = cv2.resize(template, (width, height))

    # Skip if template is larger than image
    if resized_template.shape[0] > image.shape[0] or resized_template.shape[1] > image.shape[1]:
        continue

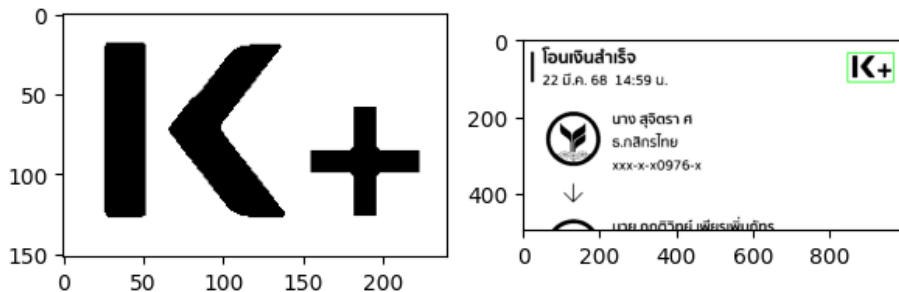
    # Perform template matching
    result = cv2.matchTemplate(image, resized_template, cv2.TM_CCOEFF_NORMED)
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)

    # Update best match if current match is better
    if max_val > best_max_val:
        best_max_val = max_val
        best_max_loc = max_loc
        best_scale = scale
        best_resized_template = resized_template
```

โค้ดการทำงานของ Function multi\_scale\_template\_matching

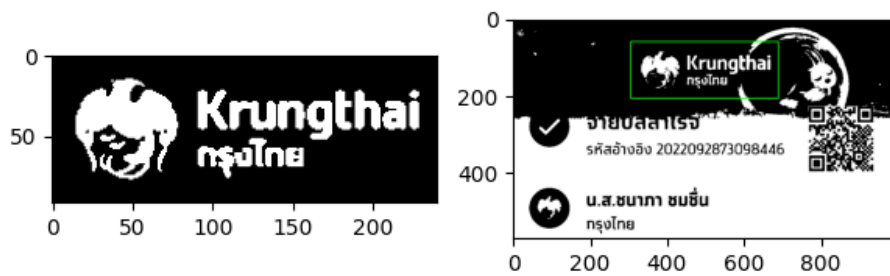
ภายในฟังก์ชันจะรับ source image, template (รูปของ object ที่ต้องการหา), และ list ของ scale แล้วนำ template มา resize ตาม scale ทั้งหมดในเทียบกับ source image เพื่อหา output เป็น scale ที่ดีที่สุด และเปอร์เซ็นต์ pixel ที่ match กัน

Template Matching  
Best scale: 0.5, Best value: 0.8990



การทำ Template Matching กับใบเสร็จอิเล็กทรอนิกส์ธนาคารกสิกร

Template Matching  
Best scale: 1.6, Best value: 0.7577



การทำ Template Matching กับใบเสร็จอิเล็กทรอนิกส์ธนาคารกรุงไทย

### 3.2 FLANN Matching

การทำ FLANN Matching จะนำรูปภาพของใบเสร็จอิเล็กทรอนิกส์ (source image) และ logo มาเทียบกัน และส่งผลลัพธ์ที่ดีที่สุดออกมาเป็นจำนวน feature ของ logo ที่ match กับ source image ทั้งหมด โดยใช้ OpenCV FlannBasedMatcher โดยสร้าง flann\_mathching() เพื่อรับข้อมูล image และ object มาประมวลผล

```
def flann_matching(image, object):
```

Function flann\_matching

```

# Initiate SIFT detector
sift = cv2.SIFT_create()

# find the keypoints and descriptors with SIFT
kp1, des1 = sift.detectAndCompute(image, None)
kp2, des2 = sift.detectAndCompute(object, None)

# FLANN parameters
FLANN_INDEX_KDTREE = 1
index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
search_params = dict(checks=50) # or pass empty dictionary

flann = cv2.FlannBasedMatcher(index_params, search_params)

matches = flann.knnMatch(des2, des1, k=2)

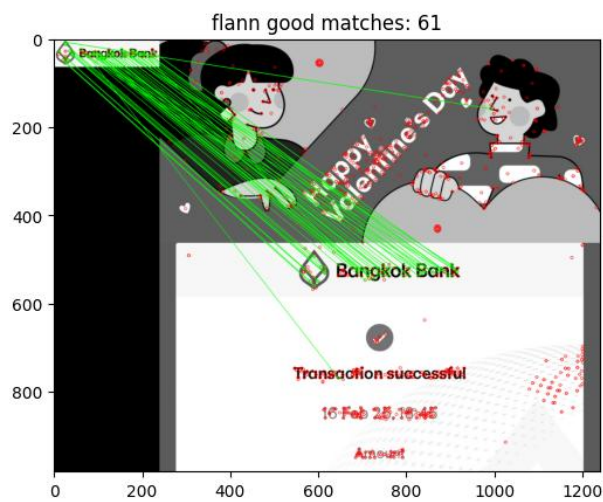
# Need to draw only good matches, so create a mask
matchesMask = [[0,0] for i in range(len(matches))]

# ratio test as per Lowe's paper
good_matches = []
for i,(m,n) in enumerate(matches):
    if m.distance < 0.7*n.distance:
        matchesMask[i]=[1,0]
        good_matches.append(m)

```

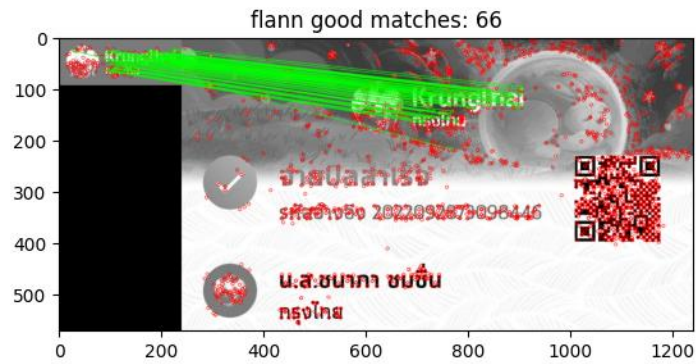
โค้ดการทำงานหลักของ Function flann\_matching

ในโค้ดจะเริ่มจากนำ algorithm Semi-supervised Incremental Feature Extraction (SIFE) มาใช้ในการหา key points และ descriptors ของ source image และ object และใช้ FLANN algorithm ในการเทียบ descriptors ของภาพทั้งสอง และคัดเฉพาะผลลัพธ์ match กันได้ดีมา โดยกำหนด threshold ไว้ที่ 0.7



การทำ FLANN Matching กับใบเสร็จอิเล็กทรอนิกส์ธนาคารกรุงเทพ





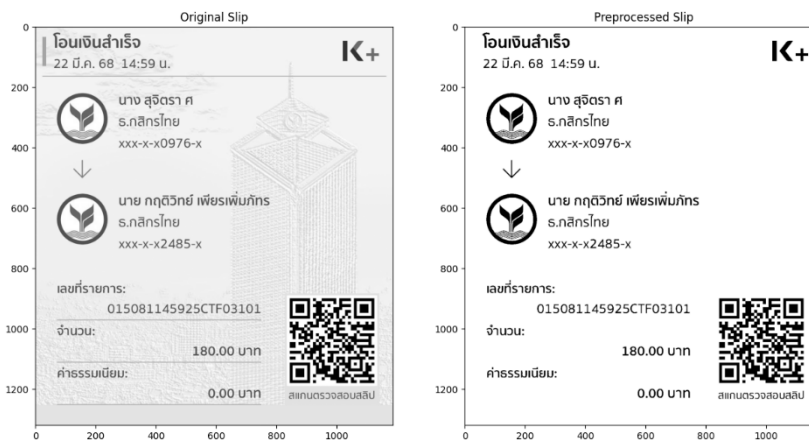
การทำ FLANN Matching กับใบเสร็จอิเล็กทรอนิกส์ธนาคารกรุงไทย

### 3.1.1.2 ขั้นตอนการทำ Optical Character Recognition (OCR)

ในการทำ OCR จะใช้ Python Tesseract OCR เพื่ออ่านตัวอักษรบนรูปภาพ มีขั้นตอนดังนี้

#### 1. Preprocessing

โดยทำ Gray scale และ Otsu thresholding เราจะไม่ใช้ Gaussian Blur ก่อนเพราะต้องการคงความคมชัดของ ตัวอักษรไว้



ตัวอย่างใบเสร็จ Gray Scale และทำ Otsu thresholding

#### 2. OCR ด้วย Tesseract OCR

สร้างฟังก์ชัน ocr\_pyesseract() รับ input เป็น path ของรูปภาพ return เป็น text ที่ตรวจจับได้ และภาพที่ใส่ bounding box ของ text แล้ว

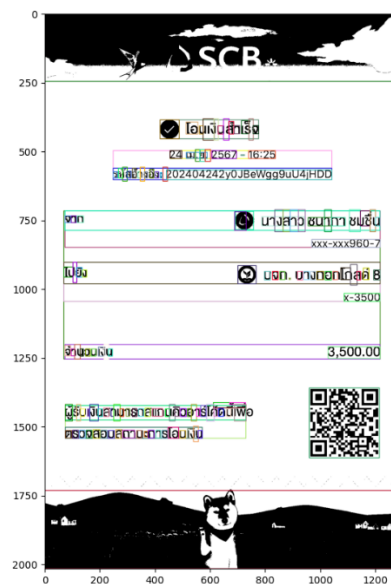
```
def ocr_pyesseract(img):
    print(img)

    img = preprocess_bank_slip(img)
    text = pytesseract.image_to_string(img, lang='tha+eng')
    d = pytesseract.image_to_data(img, lang='tha+eng', output_type=Output.DICT)

    n_boxes = len(d['level'])
    box_image = img.copy()
    box_image = cv2.cvtColor(box_image, cv2.COLOR_GRAY2BGR)
    for i in range(n_boxes):
        (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])
        # convert to rgb
        box = cv2.rectangle(box_image, (x, y), (x + w, y + h), get_random_rgb_tuple(), 2)

    return text, box
```

Function ocr\_pyesseract



ตัวอย่างผลลัพธ์ Bounding Box ที่ทำ OCR ด้วย pytesseract ของใบเสร็จอิเล็กทรอนิกส์

```
@ โอนเงินสำเร็จ

24 เม.ย. 2567 - 16:25
รหัสอ้างอิง: 202404242y0JBWgg9uU4jHDD

จาก @ นางสาว ชนาภา ชื่นชื่น
XXX-XXX960-7

ไปยัง @ บจก. บางกอกโกลด์ 8
%-3500

จำนวนเงิน 3,500.00

WO.
a

ผู้รับเงินสามารถสแกนคิวอาร์โค้ดเพื่อ
ตรวจสอบสถานะการโอนเงิน
```

ตัวอย่างผลลัพธ์ Text ที่ทำ OCR ด้วย pytesseract ของใบเสร็จอิเล็กทรอนิกส์

### 3. Text Classification ด้วย Regular Expression

โดยแบ่งฟังก์ชันสำหรับแต่ละธนาคารโดยเฉพาะเพราะแต่ละธนาคารมี Pattern ไม่เหมือนกัน โดยฟังก์ชันทั้งหมดมีดังนี้ `_extract_bangkok_info`, `_extract_krungthai_info`, `_extract_scb_info`, `_extract_kbank_info`

df[df['from\_bank'] == 'bangkok'].head(20)  
✓ 0.0s

	transaction_date	transaction_time	amount	from_account_name	from_bank	to_account_name	to_bank	ref_number
0	2025-04-05	13:03	99.0	น.ส. ร้อยแก้ว	bangkok	LINE MAN	ttb	2025040513034803056656AD\vn\kS
1	2025-04-05	14:08	15.0	น.ส. ร้อยแก้ว	bangkok	MS. Roikaew Siriwat	PromptPay	NaN
2	2025-04-05	14:09	20.0	น.ส. ร้อยแก้ว	bangkok	) MS. Roikaew Siriwat	Kasikornbank	NaN
3	2025-04-05	14:11	20.0	น.ส. ร้อยแก้ว	bangkok	ROIKAEW SIRIW	Siam Commercial Bank	2025040514111723009542008\vn\kS
4	2025-04-05	14:12	10.0	น.ส. ร้อยแก้ว	bangkok	MS. ROIKAEW SIRIWAT	Kiatnakin Phatra Bank	2025040514122924001595908\vn\kS
5	2025-04-05	14:13	12.5	น.ส. ร้อยแก้ว	bangkok	ROIKAEW SIRIWAT	Kiatnakin Phatra Bank	2025040514134124009954308\vn\kS
6	2025-04-05	14:16	3.0	น.ส. ร้อยแก้ว	bangkok	ROIKAEW SIRIWAT	Kiatnakin Phatra Bank	2025040514160724006996808\vn\kS
7	2025-04-05	14:16	3.0	NaN	bangkok	NaN	NaN	2025040514160724006996808\vn\kC
8	2025-04-05	14:13	12.5	NaN	bangkok	NaN	NaN	2025040514134124009954308\vn\kC
9	2025-04-05	14:12	NaN	NaN	bangkok	NaN	NaN	2025040514122924001595908\vn\kC
10	2025-02-16	18:45	1000.0	นาย ฤพลดา	bangkok	Mr. Kuptapa Wisarnjarusorn	Kasikornbank	2025021618455624004220508\vn\kS
11	2025-02-21	18:35	100.0	นาย ฤพลดา	bangkok	) Mr. Kuptapa Wisarnjarusorn	Kasikornbank	2025022118351223002381708\vn\kS

ตัวอย่างผลลัพธ์การทำ Text Classification จาก OCR ธนาคารกรุงเทพ

df[df['from\_bank'] == 'kbank'].tail(20)  
✓ 0.0s

	transaction_date	transaction_time	amount	from_account_name	from_bank	to_account_name	to_bank	ref_number
50	NaN	NaN	45.0	นาย ฤพลดา	kbank	NaN	NaN	01508012320160800938
51	NaN	NaN	40.0	นาย ฤพลดา	kbank	NaN	NaN	01508107570580R03813
52	NaN	NaN	50.0	นาย ฤพลดา	kbank	NaN	NaN	0150821100348PM13636
53	NaN	NaN	1500.0	นาย ฤพลดา	kbank	NaN	NaN	015083035110ATFO7232
54	NaN	NaN	30.0	นาย ฤพลดา	kbank	NaN	NaN	015084120304BTFO2854
55	NaN	NaN	50.0	นาย ฤพลดา	kbank	NaN	NaN	015084165150APP05030
56	NaN	NaN	55.0	นาย ฤพลดา	kbank	NaN	NaN	015085182100BPP02271
57	NaN	NaN	50.0	นาย ฤพลดา	kbank	NaN	NaN	015086213418APP03339
58	NaN	NaN	55.0	นาย ฤพลดา	kbank	NaN	NaN	015091113538APM04971
59	2024-08-06	20:38	127.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	014219203857APP04398
60	NaN	NaN	20.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	014224130100BPP09906
61	NaN	NaN	200.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	014281195200APP04683
62	NaN	NaN	420.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	01428418323789509030
63	NaN	NaN	25.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	014289101758APM14181
64	NaN	NaN	20.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	014291104116ATFO01180
65	NaN	NaN	520.0	น.ส. ร้อยแก้ว ศ	kbank	XXX-x-x8-x	TrueMoney Wallet	0142941315218PM03265
66	NaN	NaN	5000.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	01429914335160805106
67	2024-11-03	20:19	552.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	014308201940ATFO9112
68	NaN	NaN	40.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	015088105101APM01214
69	NaN	NaN	359.0	น.ส. ร้อยแก้ว ศ	kbank	NaN	NaN	015089220035CPP05278

ตัวอย่างผลลัพธ์การทำ Text Classification จาก OCR ธนาคารกสิกร

df[df['from\_bank'] == 'scb'].tail(20)

✓ 0.0s

	transaction_date	transaction_time	amount	from_account_name	from_bank	to_account_name	to_bank	ref_number
109	NaN	NaN	254.0	เส. ร้อยแก้ว ศิริวัฒน์	scb	NaN	NaN	202504044WAmotE67cVKQxia
110	NaN	NaN	200.0	Gas PLANET 88	scb	นางสาว ร้อยแก้ว ศิริวัฒน์	NaN	202504043
111	NaN	NaN	200.0	นางสาว ร้อยแก้ว ศิริวัฒน์	scb	NaN	NaN	202504040Pyw8FZiY2bCi1bV
112	NaN	NaN	200.0	นางสาว ชนาภา ชมชื่น	scb	NaN	NaN	202401274x4
113	2024-01-30	11:24	476.7	จ.ส.ชนาภา ชมชื่น	scb	K+ shop (311 WATSONS SALAYA MAR)	NaN	202401302ivGEnNgQ9RZLOfiq
114	2024-01-31	17:08	1500.0	นางสาว ชนาภา ชมชื่น	scb	จ.ส.ชนาภา ชมชื่น	NaN	2024011312
115	NaN	NaN	357.0	จ.ส.ชนาภา ชมชื่น	scb	K+ shop (FUJI-CENTRAL SALAYA)	NaN	202402184CHiuPUOmErMO69q8
116	NaN	NaN	890.0	นางสาว ชนาภา ชมชื่น	scb	จ.ส.ชนาภา ชมชื่น	NaN	20240222190DW7pXn8T3XR6a
117	NaN	NaN	200.0	นางสาว ชนาภา ชมชื่น	scb	NaN	NaN	202403181JCGlbJp2LvwX19jd
118	NaN	NaN	1088.0	ส.ชนาภา ชมชื่น	scb	K+ shop (MK RESTAURANT GROUP)	NaN	2024041331PIYCAVNVKQJRLVA
119	NaN	NaN	2000.0	นางสาว ชนาภา ชมชื่น	scb	© บริษัท ห้างทอง ทองไทย โลตัสสาขาบุรี จำกัด	NaN	2024041802hVtaEKgGKAERu
120	NaN	NaN	3500.0	นางสาว ชนาภา ชมชื่น	scb	บจก. บางกอกโกลด์ 8	NaN	202404242yOjBeWgg9uJ4jHDD
121	2024-05-02	20:48	164.0	นางสาว ชนาภา ชมชื่น	scb	นายวรเชษฐ์ จัญญะดี	NaN	2024050235BuGyWxn488XF5xV
122	NaN	NaN	NaN	NaN	scb	NaN	NaN	202402114wO2HWd76yOccOJKQ
123	NaN	NaN	34.0	wens ชนาภา ชมชื่น	scb	จ.ส.ชนาภา ชมชื่น	NaN	202404140PFqkbelPA4ULAHd
124	2024-05-08	18:45	1429.0	จ.ส.ชนาภา ชมชื่น	scb	K+ shop (BNN RESTAURANT-	NaN	202405083eddxRmp6SzRb4B2
125	2024-05-02	20:49	10.0	นางสาว ชนาภา ชมชื่น	scb	ป๋ิ นายวรเชษฐ์ จัญญะดี	NaN	202405020e2yk1bF7Ly3M0t
126	2024-05-03	12:37	55.0	ส.ชนาภา ชมชื่น	scb	น.ส. ปวีจิน จอมแบ่ง	PromptPay	2024050341
127	2024-08-15	08:04	100.0	เส. ร้อยแก้ว ศิริวัฒน์	scb	NaN	NaN	202408151gcrSyPX3umYtpuH1
128	2024-10-25	14:34	5000.0	ส.ร้อยแก้ว ศิริวัฒน์	scb	NaN	NaN	202410250aamSEyL6EsOVheE

ตัวอย่างผลลัพธ์การทำ Text Classification จาก OCR ธนาคารไทยพาณิชย์

df[df['from\_bank'] == 'krungthai'].head(20)

✓ 0.0s

	transaction_date	transaction_time	amount	from_account_name	from_bank	to_account_name	to_bank	ref_number
70	NaN	NaN	20.00	usunshwdi et* * *	krungthai	น.ส. ชรินทร์ทิพย์ ศิริวัฒน์	NaN	Ab14539fe68ae42f5
71	NaN	NaN	20.00	usunshwdi et* * *	krungthai	น.ส. ชรินทร์ทิพย์ ศิริวัฒน์	NaN	Ab14539fe68ae42f5
72	NaN	NaN	300.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
73	2022-05-05	01:29	550.00	น.ส.ชนาภา ชมชื่น	krungthai	นาย คุปดาภา วิสารจากร	NaN	2022050587805714
74	NaN	NaN	1000.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
75	NaN	NaN	200.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
76	NaN	NaN	75.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
77	NaN	NaN	100.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	2022072570728520
78	NaN	NaN	1800.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
79	NaN	NaN	150.00	iG	krungthai	NaN	NaN	1521400331020220901
80	NaN	NaN	0.00	Oe Krunatt ai	krungthai	NaN	NaN	NaN
81	NaN	NaN	120.00	Oe Krunatt ai	krungthai	NaN	NaN	NaN
82	NaN	NaN	33.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
83	NaN	NaN	20.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
84	2022-10-10	00:49	396.00	น.ส.ชนาภา ชมชื่น	krungthai	SHOPEEPAY (THAILAND)	ShopeePay	1 SHPK5KZGK9P
85	2022-10-12	14:55	320.34	น.ส.ชนาภา ชมชื่น	krungthai	MPAY	MPAY	2022101298737736
86	2022-10-12	16:34	104.12	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	NaN
87	2022-11-01	08:12	150.00	น.ส.ชนาภา ชมชื่น	krungthai	NaN	NaN	2022110120046516

ตัวอย่างผลลัพธ์การทำ Text Classification จาก OCR ธนาคารกรุงไทย

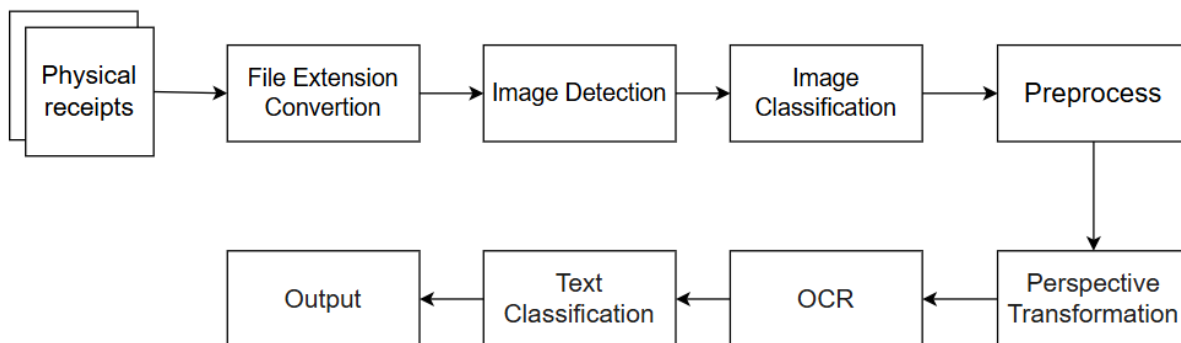
3.1.2 โครงสร้างภาพรวมของระบบ

โดยภาพรวมจะให้ user อัปโหลดรูปภาพของใบเสร็จจ้อเล็กทรอนิกส์และระบบจะ preprocess รูปภาพเพื่อนำมาทำ Template Matching และ FLANN Matching เพื่อแยกว่าเป็นธนาคารอะไร หลังจากนั้นจะเข้าสู่การทำ OCR ได้ Text ออกมาทำ Regular Expression ของแต่ละธนาคาร แล้วสรุปออกมาเป็นตาราง

## 3.2 ภาพรวมและสภาพแวดล้อมของการพัฒนาระบบสกัดข้อมูลสำหรับใบเสร็จการทำธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ (ATM)

### 3.2.1 วิธีการพัฒนา

วิธีการพัฒนาระบบสามารถแบ่งออกได้เป็น 2 ขั้นตอนหลัก คือ ขั้นตอนการตรวจจับพื้นที่ของใบเสร็จบนภาพ และขั้นตอนการทำ OCR โดยมีขั้นตอนย่อยดังภาพ



ภาพรวมการพัฒนาระบบในส่วนใบเสร็จการทำธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ

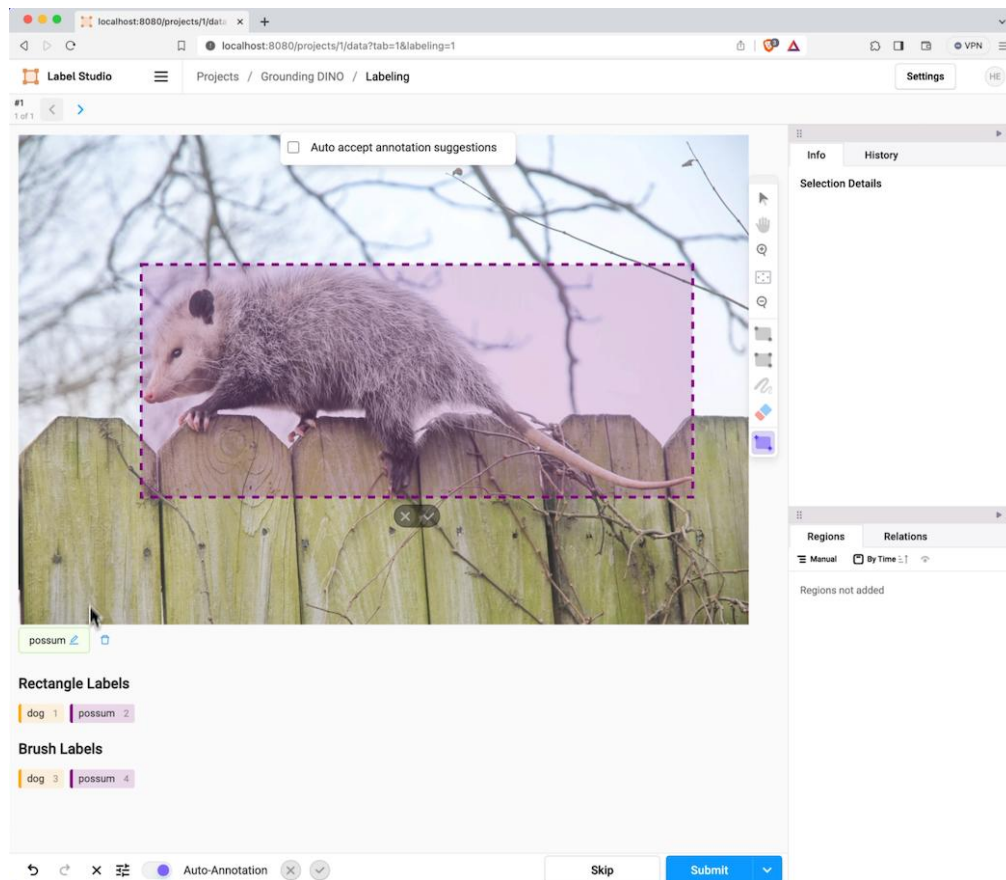
#### 3.2.1.1 ขั้นตอนการตรวจจับพื้นที่ใบเสร็จบนภาพ

##### 1. กระบวนการเตรียมงานก่อนขั้นตอนการตรวจจับพื้นที่ใบเสร็จบนภาพ

ก่อนที่จะเริ่มกล่าวถึงการขั้นตอนตรวจจับพื้นที่ใบเสร็จบนภาพ ขั้นตอนการ Fine-tuned YOLO model จะต้องถูกกล่าวถึงก่อน เนื่องจากการ Fine-tuned ด้วย dataset ของโครงการเอง อีกทั้งยังเป็นส่วนสำคัญของโครงการอีกด้วย

Dataset ถูกเก็บเองโดยสมาชิกภาพในกลุ่ม ภายใน Dataset ประกอบด้วยภาพใบเสร็จจาก 4 ธนาคาร ได้แก่ ธนาคารกรุงเทพ ธนาคารกรุงไทย ธนาคารกสิกรไทย และธนาคารไทยพาณิชย์ โดย Dataset ที่เก็บได้มาจากการถ่ายภาพโทรศัพท์มือถือจากใบเสร็จตัวจริง เว้นแต่ธนาคารไทยพาณิชย์ที่มีอุปสรรคในการหาเป็นอย่างมาก จึงตัดสินใจนำภาพจาก Internet มา Fine-tuned model ซึ่งอาจส่งผลต่อประสิทธิภาพการทำงาน ซึ่งภาพทั้งหมดที่นำมาจาก Internet ล้วนแล้วแต่นำมาใช้เพื่อการศึกษาทั้งสิ้น ไม่มีการนำไปใช้ในเชิงพาณิชย์แต่อย่างใด โดยภาพ Dataset ที่สมาชิกในกลุ่มหาเอง ถูกออกแบบให้อยู่ในหลากหลายสภาพแวดล้อม เช่น พื้นหลังเรียบ พื้นหลังลาย แสงมาก แสงน้อย ใบเสร็จยับมาก ใบเสร็จยับน้อย ใบเสร็จเรียบ เพื่อให้โมเดลมีประสิทธิภาพมากที่สุด

หลังจากเก็บ Dataset แล้วจึงนำไป label เพื่อระบุพื้นที่ของใบเสร็จ และธนาคารเจ้าของใบเสร็จ หรือสามารถเรียกอีกอย่างได้ว่า class ของใบเสร็จ โดยใช้โปรแกรม Label Studio



ภาพตัวอย่างการใช้งานโปรแกรม Label Studio ในการ label ภาพและกำหนด class

อ้างอิง: <https://labelstud.io/blog/using-text-prompts-for-image-annotation-with-grounding-dino-and-label-studio/>

หลังจากนั้นจะเริ่มในส่วนของการ train YOLO model โดยเริ่มจากการสร้างไฟล์ YAML ที่เป็น Configuration file เหมือนเป็นไฟล์ที่บอกตำแหน่งของ dataset ที่ใช้ train และ test กับ YOLO และบอก class name ด้วย

```

import yaml
import os

def create_data_yaml(path_to_classes_txt, path_to_data_yaml):

    # Read class.txt to get class names
    if not os.path.exists(path_to_classes_txt):
        print(f'classes.txt file not found. Please create a classes.txt labelmap and move it to {path_to_classes_txt}')
        return
    with open(path_to_classes_txt, 'r') as f:
        classes = []
        for line in f.readlines():
            if len(line.strip()) == 0: continue
            classes.append(line.strip())
        number_of_classes = len(classes)

    # Create data dictionary
    data = {
        'path': r'C:\academic\uni\4term2\cs381\term project\data',
        'train': r'C:\academic\uni\4term2\cs381\term project\data\train\images',
        'val': r'C:\academic\uni\4term2\cs381\term project\data\test\images',
        'nc': number_of_classes,
        'names': classes
    }

    # Write data to YAML file
    with open(path_to_data_yaml, 'w') as f:
        yaml.dump(data, f, sort_keys=False, default_flow_style=True)
    print(f'Created config file at {path_to_data_yaml}')

    return

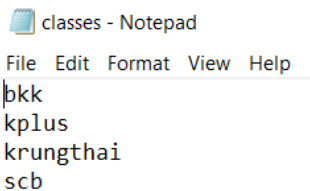
# Define path to classes.txt and run function
path_to_classes_txt = r'C:\academic\uni\4term2\cs381\term project\data\train\classes.txt'
path_to_data_yaml = r'C:\academic\uni\4term2\cs381\term project\data.yaml'

create_data_yaml(path_to_classes_txt, path_to_data_yaml)

print('\nFile contents:\n')

```

ภาพการโปรแกรมการสร้างไฟล์ YAML

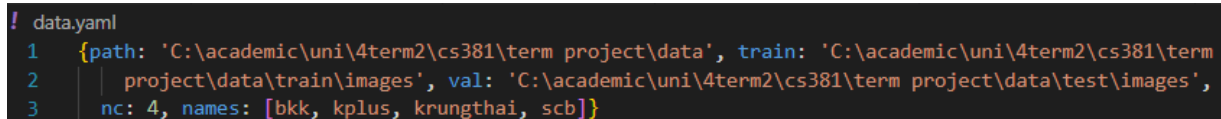


```

classes - Notepad
File Edit Format View Help
bkk
kplus
krungthai
scb

```

ภาพข้อมูลในไฟล์ classes.txt



```

! data.yaml
1 {path: 'C:\academic\uni\4term2\cs381\term project\data', train: 'C:\academic\uni\4term2\cs381\term
2   project\data\train\images', val: 'C:\academic\uni\4term2\cs381\term project\data\test\images',
3   nc: 4, names: [bkk, kplus, krungthai, scb]}

```

ภาพผลลัพธ์ของไฟล์ data.yaml ที่จะถูกนำไปใช้กับ YOLO ในขั้นตอนการ train model ต่อไป

จากนั้นนำ data.yaml ที่ได้ไปใช้ในการ train YOLO model

```
Train

# model training
!yolo detect train data-r 'data.yaml' model=yolov8s.pt epochs=60 imgsz=640

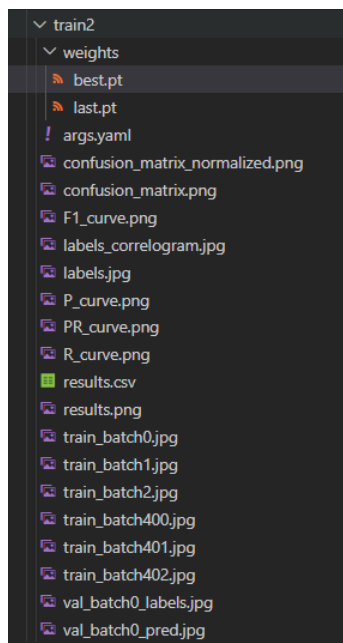
New https://pypi.org/project/ultralytics/8.3.124 available 🟡 Update with 'pip install -U ultralytics'
Ultralytics 8.3.109 Python-3.13.2 torch-2.6.0+cpu CPU (Intel Core(TM) i5-8265U 1.60GHz)
engine/trainer: task=detect, mode=train, model=yolov8s.pt, data=data.yaml, epochs=60, time=None, patience=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=None, workers
Downloading https://ultralytics.com/assets/Arial.ttf to 'C:\Users\User\AppData\Roaming\Ultralytics\Arial.ttf'...
Overriding model.yaml nc=80 with nc=4

   from  n  params module arguments
   --  --  --  --  --
0      -1  1    928 ultralytics.nn.modules.conv.Conv [3, 32, 3, 2]
1      -1  1  18560 ultralytics.nn.modules.conv.Conv [32, 64, 3, 2]
2      -1  1  29056 ultralytics.nn.modules.block.C2f [64, 64, 1, True]
3      -1  1  73984 ultralytics.nn.modules.conv.Conv [64, 128, 3, 2]
4      -1  2  197632 ultralytics.nn.modules.block.C2f [128, 128, 2, True]
5      -1  1  295424 ultralytics.nn.modules.conv.Conv [128, 256, 3, 2]
6      -1  2  788480 ultralytics.nn.modules.block.C2f [256, 256, 2, True]
7      -1  1  1180672 ultralytics.nn.modules.conv.Conv [256, 512, 3, 2]
8      -1  1  1838080 ultralytics.nn.modules.block.C2f [512, 512, 1, True]
9      -1  1  656896 ultralytics.nn.modules.block.SPPF [512, 512, 5]
10     -1  1      0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
11     [-1, 6] 1      0 ultralytics.nn.modules.conv.Concat [1]
12     -1  1  591360 ultralytics.nn.modules.block.C2f [768, 256, 1]
13     -1  1      0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
14     [-1, 4] 1      0 ultralytics.nn.modules.conv.Concat [1]
15     -1  1  148224 ultralytics.nn.modules.block.C2f [384, 128, 1]
16     -1  1  147712 ultralytics.nn.modules.conv.Conv [128, 128, 3, 2]
17     [-1, 12] 1      0 ultralytics.nn.modules.conv.Concat [1]
...

scb      5      5    0.992      1    0.995    0.995
Speed: 2.1ms preprocess, 312.6ms inference, 0.0ms loss, 1.2ms postprocess per image
Results saved to runs\detect\train2
🔗 Learn more at https://docs.ultralytics.com/modes/train
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

ภาพการ train YOLO model

ภาพด้านล่างเป็นผลลัพธ์จากการ train YOLO model จากนั้นเราจะได้ไฟล์ best.pt ซึ่งเป็นไฟล์โมเดลที่มีความเก่งที่สุดไปใช้ในขั้นตอนถัดไป



ภาพผลลัพธ์จากการ train YOLO model



## 2. ขั้นตอนตรวจสอบประเภทของไฟล์รูปภาพ

หลังจากรับภาพถ่ายที่ตรงตามข้อกำหนดของระบบเข้ามาแล้ว ต้องตรวจสอบประเภทไฟล์ของภาพก่อน เนื่องจาก OpenCV สามารถรับประเภทไฟล์ภาพจำกัด อาทิ jpg, jpeg, png, bmp อย่างไรก็ตาม ภาพถ่ายจากโทรศัพท์มือถือยี่ห้อ Apple ส่งออกภาพที่มีไฟล์ประเภท heif และ heic จึงต้องมีการตรวจสอบและแปลงประเภทของไฟล์ภาพก่อน

```
# supported filetype
img_ext_list = ['.jpg', '.JPG', '.jpeg', '.JPEG', '.png', '.PNG', '.bmp', '.BMP']

# check image type
file_extension = os.path.splitext(image_path)[1]
# change image type(if .heif/.heic)
if file_extension in img_ext_list:
    # read image
    image = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)
elif (file_extension=='.HEIC'or file_extension=='.heic'or file_extension=='.HEIF'or file_extension=='.heif'):
    # convert to .JPEG
    image = convert_heic_or_heif_to_jpeg(image_path) # return as np array
    if image is None:
        print('ERROR: HEIC/HEIF file conversion failed. Please check the file.')
        sys.exit(0)
else:
    print(f'Input {image_path} is invalid. Please try again.')
    sys.exit(0)
```

### ภาพการโปรแกรมตรวจสอบประเภทของไฟล์

การแปลงประเภทของไฟล์ คือ การแปลง color scale เดิมให้เป็น color scale ของประเภทไฟล์ที่เราต้องการ ในที่นี้คือต้องการแปลงให้เป็นไฟล์ประเภท jpeg ที่มี color scale คือ RGB จากนั้นส่งกลับไปเก็บที่ตัวแปรตั้งต้นในรูปแบบของ numpy array ทำให้ไม่ต้องบันทึกไฟล์ที่แปลงแล้วลงเครื่อง

```
# Register HEIC support
pillow_heif.register_heif_opener()
def convert_heic_or_heif_to_jpeg(filepath):
    try:
        img = Image.open(filepath).convert("RGB")
        return np.array(img) # Return image as NumPy array
    except Exception as e:
        return None
```

### ภาพการโปรแกรมแปลงประเภทของไฟล์

## 3. ขั้นตอนการตรวจหาใบเสร็จและระบุประเภทธนาคารเจ้าของใบเสร็จ

ภาพจะถูกนำเข้าโมเดล YOLO เพื่อทำการตรวจจับพื้นที่ใบเสร็จในภาพ และทำการ classify ประเภทของใบเสร็จบนภาพ อีกทั้งยังสร้าง Region of Interest (ROI) ด้วยการ crop ภาพ โดยใช้ตำแหน่งของ bounding box เพื่อเป็นตัวช่วยในขั้นตอน Perspective Transformation

```

# load YOLO model and get labelmap
model = YOLO(model_path, task='detect') # task='detect' => obj detection => get bb and class
labels = model.names # is to map between class id and classname

# resize for appropriate scale
image = resize(image)

# rescale color scale -> effect model's confidence percentage
image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

# -----image preprocessing-----
# run detection
results = model(image)
# get the results
detections = results[0].boxes

# project scope => 1 image = 1 receipt
for i in range(0,1): # if there are several obj, get the first one
    # get bounding box coordinates
    # Ultralytics returns results in Tensor format, which have to be converted to a regular Python array
    xyxy_tensor = detections[i].xyxy.cpu() # Detections in Tensor format in CPU memory
    xyxy = xyxy_tensor.numpy().squeeze() # Convert tensors to Numpy array
    xmin, ymin, xmax, ymax = xyxy.astype(int) # Extract individual coordinates and convert to int

    # get classname
    classidx = int(detections[i].cls.item())
    classname = labels[classidx] # map to get classname of the obj

    # get confidence
    conf = detections[i].conf.item()

    # cropped
    # roi = img[y1:y2,x1:x2]
    after_yolo = image[ymin:ymax,xmin:xmax]

```

### ภาพโปรแกรมการใช้งานโมเดล Yolo

เนื่องจากจากการทดลองทำไปสักระยะพบว่า ไฟล์ที่มาจากโทรศัพท์ APPLE มีขนาดใหญ่มาก ทำให้หลังจากออกจาก YOLO model แล้วภาพที่ได้ จะได้ไม่เต็มภาพ จึงต้องมีการแปลงขนาดของภาพก่อนนำเข้า YOLO model

```

# resize image for display
def resize(image, max_width=1280, max_height=720):
    height, width = image.shape[:2]
    scale = min(max_width / width, max_height / height)
    if scale < 1.0:
        image = cv2.resize(image, (int(width * scale), int(height * scale)))
    return image

```

### ภาพการโปรแกรม function resize

#### 4. ขั้นตอนการเตรียมภาพก่อนขั้นตอน Perspective Transformation

ก่อนการนำเข้าไปทำ Perspective Transformation ควรจะต้อง Preprocess ก่อนนำเข้าไป เพื่อเน้นให้ขอบชัดขึ้น ทำให้ประสิทธิภาพในการหามุมไบเอร์ทั้ง 4 มุมดีขึ้น

```
# ---perspective transformation---
# edge detection
gray_image = cv2.cvtColor(after_yolo, cv2.COLOR_BGR2GRAY)

# get 4 coordinates for perspective transformation
# optimize image before go get 4 coordinates
blurred = cv2.GaussianBlur(gray_image, (3,3),0)
thresh = cv2.adaptiveThreshold(blurred, 255,
                               cv2.ADAPTIVE_THRESH_MEAN_C,
                               cv2.THRESH_BINARY_INV, 21, 10)
_, binaryInv = cv2.threshold(thresh, 0, 255, cv2.THRESH_BINARY_INV)
```

ภาพโปรแกรมการ Preprocess ภาพก่อนขั้นตอน Perspective Transformation

เหตุผลที่เลือกใช้ adaptive thresholding แล้วค่อยมาทำ binary inverse ที่หลังแทนการทำ binary ไปเลย เพราะว่าจากการทดลองการใช้ binary ทำให้ไม่สามารถจัดการกับเงาบนภาพได้ จะกลายเป็นแทบดำไปเลย ทำให้เกิดข้อจำกัดของโครงการที่เพิ่มมากขึ้น จึงเปลี่ยนไปใช้ adaptive thresholding ร่วมกับ binary inverse แทน ซึ่งสามารถจัดการกับภาพที่มีเงาเข้มในระดับหนึ่งได้ดีมากกว่า แต่อย่างไรก็ตามยังคงมีข้อเสีย การใช้ร่วมกันของ 2 เทคนิคดังกล่าวไม่สามารถจัดการกับไบเอร์ที่ขยับเหมือนอย่าง binary ทำได้ ซึ่งจะส่งผลต่อคุณภาพของการสกัดข้อความจากภาพ (OCR)



ภาพต้นฉบับ, ภาพที่ถูกปรับปรุงด้วย binary, ภาพที่ถูกปรับปรุงด้วย adaptive thresholding และ binary inverse ตามลำดับ

## 5. ขั้นตอนการทำ Perspective Transformation

ขั้นตอนนี้ประกอบด้วย 2 ส่วน นั่นคือ ขั้นตอนการหามุมทั้ง 4 ของใบเสร็จ และขั้นตอนการเชื่อมมุมเก่าไปเป็นมุมใหม่

ขั้นตอนแรก การหามุมทั้ง 4 โดยนำภาพที่ Preprocess แล้วเข้ามาทำ โดยจะใช้ OpenCV หาเส้นกรอบภาพที่ใหญ่ที่สุด แล้วทำลูปไปเรื่อย ๆ จนกว่าจะได้มุมของใบเสร็จทั้ง 4 มุม แล้วนำมุมทั้ง 4 ที่ได้มาจัดให้เรียงในลำดับ ซ้ายบน, ขวาบน, ซ้ายล่าง, และขวาล่าง เพื่อที่จะนำไปใช้ในการเชื่อมมุมในขั้นตอนถัดไปได้สะดวกขึ้น

```
# get 4 coordinates
# corners' order=TL, TR, BL, BR
corners, detected_4_coor_with_contour = get_4_coordinates(binaryInv,gray_image) #binary
for i,(x,y) in enumerate(corners):
    print(f"Corner {i + 1}: (x={x}, y={y})")
```

```

def get_4_coordinates(binary, gray_image):
    # find the largest contour in the threshold image
    cnts = cv2.findContours(binary.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)
    c = max(cnts, key=cv2.contourArea)
    (x, y, w, h) = cv2.boundingRect(c)

    # to demonstrate the impact of contour approximation -> loop over a number of epsilon sizes
    for eps in np.linspace(0.001, 0.05, 10):
        # approximate the contour
        peri = cv2.arcLength(c, True)
        approx = cv2.approxPolyDP(c, eps * peri, True)

        # draw the approximated contour on the image
        detected_4_coor_with_contour = gray_image.copy()
        cv2.drawContours(detected_4_coor_with_contour, [approx], -1, (0, 255, 0), 3)
        text = "eps={:.4f}, num_pts={}".format(eps, len(approx))
        cv2.putText(detected_4_coor_with_contour, text, (x, y - 15), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

        if len(approx)==4:
            break

    if len(approx)==4:
        corners = approx.reshape(4, 2) # reshape to (4, 2) for convenience
        for i, point in enumerate(corners):
            x, y = point

        # sort the corners for perspective transform
        # Use top-left, top-right, bottom-right, bottom-left order
        def sort_corners(pts):
            pts = pts[np.argsort(pts[:, 1])] # sort by y (top to bottom)
            top, bottom = pts[:2], pts[2:]
            # sort top by x (left to right)
            top = top[np.argsort(top[:, 0])]
            # sort bottom by x (left to right, not reversed anymore)
            bottom = bottom[np.argsort(bottom[:, 0])]
            # order: TL, TR, BL, BR
            return np.array([top[0], top[1], bottom[0], bottom[1]])

        ordered_corners = sort_corners(corners)
    else:
        print(f"ERROR: Got {len(approx)} points - not a quadrilateral. Try adjusting epsilon.")
        sys.exit(0)

    return ordered_corners, detected_4_coor_with_contour

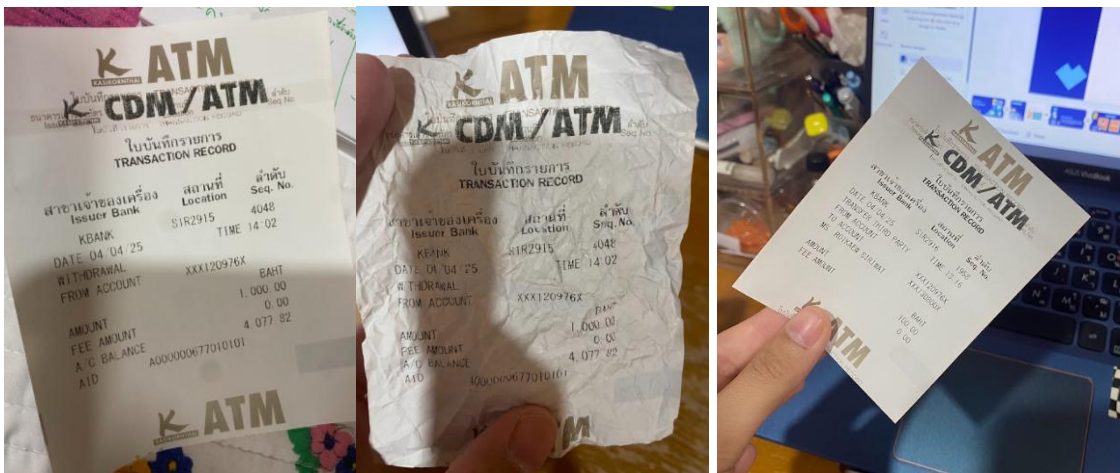
```

ภาพการโปรแกรมหามุมทั้ง 4 ของใบเสร็จในภาพ



ภาพผลลัพธ์ของการหามุมทั้ง 4 ของใบเสร็จ (กรอบสีดำ)

อย่างไรก็ตาม การโปรแกรมดังกล่าวมีข้อเสียใหญ่อยู่หนึ่งข้อ คือ ถ้ามีจำนวนรูปจนครบรอบแล้วแต่มีมากกว่าหรือน้อยกว่า 4 มันจะทำให้ขั้นตอนต่อไปไม่ได้ แล้วโปรแกรมจะต้องจบลงเลย ซึ่งในถ้าภาพถูก Preprocess ด้วย binary จะเกิดคอขวดที่แคบกว่าภาพที่ถูก Preprocess ด้วย adaptive thresholding ร่วมกับ binary inverse นอกจากนี้จากการสังเกต ภาพที่เกิดปัญหาในขั้นตอนนี้นั้นมักเป็นภาพถ่ายใบเสร็จไม่เต็มใบ ภาพใบเสร็จยับมาก ภาพที่มีสีพื้นหลังใกล้เคียงกับใบเสร็จ และภาพใบเสร็จที่บิดหมุนเอียงจนทำให้เห็นภาพพื้นหลังลาย ๆ มากกว่าที่ควรจะเป็น



ภาพตัวอย่างภาพถ่ายใบเสร็จไม่เต็มใบ ภาพใบเสร็จยับมาก และภาพใบเสร็จที่บิดหมุนเอียงจนทำให้เห็นภาพพื้นหลังลาย ๆ มากกว่าที่ควรจะเป็น ตามลำดับ

```
0: 640x480 1 kplus, 288.9ms
Speed: 7.9ms preprocess, 288.9ms inference, 1.2ms postprocess per image at shape (1, 3, 640, 480)
ERROR: Got 6 points – not a quadrilateral. Try adjusting epsilon.

An exception has occurred, use %tb to see the full traceback.

SystemExit: 0
```

ภาพตัวอย่าง error ที่เกิดขึ้น

### 3.2.1.2 ขั้นตอนการทำ OCR

#### 1. ขั้นตอนการสกัดข้อความจากภาพ

หลังจากได้ภาพที่ทำ Perspective transformation เพื่อเพิ่มประสิทธิภาพให้กับเครื่องมือ OCR อย่าง Pytesseract แล้วก็นำมาทำ OCR โดยภาพที่นำเข้ามาทำ OCR เป็นภาพ Perspective Transformation ที่ผ่านการ Preprocess อีกครั้งก่อนเข้าไปทำ OCR เพื่อเพิ่ม contrast ระหว่างตัวอักษรกับพื้นหลัง

เหตุผลที่เลือกใช้งาน adaptive thresholding ร่วมกับ binary inverse เป็นเพราะถ้าใช้ภาพ grayscale กับการทำ OCR และถ้ามีเงาที่เข้มระดับหนึ่ง OCR จะไม่สามารถอ่านข้อความบริเวณที่มีเงาพาดได้เลยสักตัวอักษรเดียว การใช้ adaptive thresholding ร่วมกับ binary inverse สามารถแก้ไขในส่วนนี้ได้ แต่อย่างไรก็ตาม การใช้ 2 เทคนิคดังกล่าวมีข้อเสียอย่างที่ได้กล่าวไปแล้วในข้างต้น ทำให้เกิด noise เยอะมาก แต่ก็ยังพอสกัดข้อความออกมาได้บ้าง แม้จะไม่ถูกต้องก็ตาม

```
thresh = cv2.adaptiveThreshold(perspective_trans, 255,
                               cv2.ADAPTIVE_THRESH_MEAN_C,
                               cv2.THRESH_BINARY_INV, 21, 10)
_, binaryInv = cv2.threshold(thresh, 0, 255, cv2.THRESH_BINARY_INV)

# -----OCR-----
# pytesseract
custom_config = r'--oem 3 --psm 6'
textPytess = pytesseract.image_to_string(binaryInv, lang='tha+eng', config=custom_config)
print("-----PYTESSERACT-----")
print(textPytess)

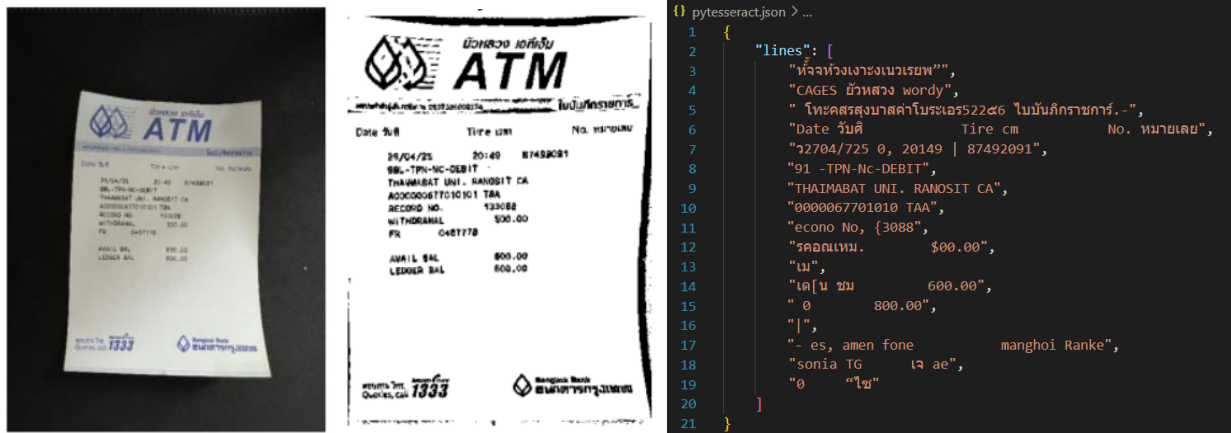
lines = textPytess.splitlines()

with open("pytesseract.json", "w", encoding="utf-8") as f:
    json.dump({"lines": lines}, f, ensure_ascii=False, indent=4)
```

ภาพการใช้งาน OCR ชื่อ Pytesseract

จากนั้นจะได้ผลลัพธ์ในรูปแบบไฟล์ json





ภาพต้นฉบับ ภาพที่ผ่านการ Perspective Transformation, adaptive thresholding, binary inverse ก่อนการทำ OCR และผลลัพธ์ที่ได้หลังจากการทำ OCR ตามลำดับ

## 2. ขั้นตอนการทำ Text Classification

ในขั้นตอนตอน จะใช้ Regular expression ในการคัดแยกข้อมูลที่เราต้องการจากข้อมูลที่สกัดได้ทั้งหมด โดยจะแยกเป็นของแต่ละธนาคารเลย อย่างไรก็ตาม ผู้จัดทำไม่สามารถทำ Regular expression ของธนาคารไทยพาณิชย์ได้ เนื่องจาก dataset ที่ไม่เสถียร แม้จะเป็นใบเสร็จจากการทำธุรกรรมแบบเดียวกัน แต่การจัดวางตัวหนังสือต่างกัน จึงขอละการทำ Text classification สำหรับธนาคารไทยพาณิชย์ จะแสดงเป็นข้อมูลที่สกัดได้จาก OCR เท่านั้น

```
if classname=='bkk':
    extracted_text = bkk_extracted(lines)

if classname=='kplus':
    extracted_text = kplus_extracted(lines)

if classname=='krungthai':
    extracted_text = krungthai_extracted(lines)

if classname=='scb':
    extracted_text = lines
```

ภาพโปรแกรมการทำ Text Classification แยกตามธนาคาร

การทำ Text Classification ของธนาคารกรุงเทพ ข้อมูลที่ต้องการสกัดจากใบเสร็จธนาคารกรุงเทพ มีดังนี้ วันที่, เวลา, ประเภทการทำธุรกรรม, จำนวนเงิน, และจำนวนเงินคงเหลือที่ใช้ได้



```

def bkk_extracted(lines):
    def clean_text(text):
        return text.replace('0', '0').replace('o', '0').replace('I', '1').replace('l', '1').replace(',', '').replace('#', '').strip()

    # Regex patterns
    date_pattern = r"\b\d{2}/\d{2}/\d{2}\b"
    time_pattern = r"\b\d{2}:\d{2}\b"
    withdrawal_pattern = r"\bWITHDRAWAL\b"
    amount_pattern = r"\b\d{1,3}(?:,\d{3})*(?:\.\d{2})\b"
    avail_bal_label_pattern = r"AVAIL\s+BAL"

    extracted = {
        "date": None,
        "time": None,
        "transaction_type": None,
        "withdrawal_amount": None,
        "available_balance": None
    }

    for i, raw_line in enumerate(lines):
        line = clean_text(raw_line.upper())

        # Extract date
        if not extracted["date"]:
            match = re.search(date_pattern, line)
            if match:
                extracted["date"] = match.group()

        # Extract time
        if not extracted["time"]:
            match = re.search(time_pattern, line)
            if match:
                extracted["time"] = match.group()

        # Extract transaction type
        if not extracted["transaction_type"]:
            if re.search(withdrawal_pattern, line):
                extracted["transaction_type"] = "WITHDRAWAL"

        # Extract withdrawal amount (the line usually contains the word WITHDRAWAL)
        if extracted["transaction_type"] == "WITHDRAWAL" and not extracted["withdrawal_amount"]:
            if "WITHDRAWAL" in line and i+1 < len(lines):
                # Check current or next line for amount
                current_line_match = re.search(amount_pattern, line)
                next_line_match = re.search(amount_pattern, clean_text(lines[i+1].upper()))
                if current_line_match:
                    extracted["withdrawal_amount"] = current_line_match.group()
                elif next_line_match:
                    extracted["withdrawal_amount"] = next_line_match.group()

        # Extract available balance
        if re.search(avail_bal_label_pattern, line):
            match = re.search(amount_pattern, line)
            if match:
                extracted["available_balance"] = match.group()
            elif i + 1 < len(lines):
                # Try next line if not found on current
                next_match = re.search(amount_pattern, clean_text(lines[i + 1].upper()))
                if next_match:
                    extracted["available_balance"] = next_match.group()

    return extracted

```

ภาพการโปรแกรม Text Classification ของธนาคารกรุงเทพ



ภาพที่ใช้ในการทำ OCR ของธนาคารกรุงไทย

```

() pyesseract.json > [ ] lines > 12
1 {
2   "lines": [
3     ". eg นลค พ 0",
4     "OS: ขั้วหลวง fide",
5     "ขี-- "ae -",
6     "mms bday อรวิธตรงรงรต .ไปบันทึกรายการ ..",
7     "Data วันที่ Time เลง Na, พบเซเลย",
8     "' 29/04/25 20150 87499097",
9     ". 2B, -THN-We-DEBIT",
10    "THAMMASAT UNL. RANOBIT 06",
11    "4000006771610151 TAA",
12    "คชชอลด No. 133055",
13    "พเรทน. 200,00",
14    "เลา",
15    "เ 490.00",
16    "เรชช mag 400.00",
17    "ergo flee ๑ Bangkok Beak",
18    "wins,",
19    "สัสชีอิด ธนคทพารพรงเดพน"
20  ]
21 }

```

ผลลัพธ์ของการทำ OCR ของธนาคารกรุงไทย

```

*****
{
  "date": "29/04/25",
  "time": null,
  "transaction_type": null,
  "withdrawal_amount": null,
  "available_balance": null
}
*****

```

ผลลัพธ์ของการทำ Text Classification ของธนาคารกรุงไทย

การทำ Text Classification ของธนาคารกสิกร ข้อมูลที่ต้องการสกัดจากใบเสร็จธนาคารกสิกรไทย มีดังนี้  
วันที่, เวลา, ประเภทการทำธุรกรรม, เลขบัญชีธนาคารต้นทาง, จำนวนเงิน, ค่าธรรมเนียม,  
และจำนวนเงินคงเหลือที่ใช้ได้

```
def kplus_extracted(lines):  
    extracted_data = {  
        "date": None,  
        "time": None,  
        "transaction_type": None,  
        "from_account": None,  
        "withdrawal_amount": None,  
        "fee_amount": None,  
        "account_balance": None  
    }  
  
    date_pattern = r"DATE\s*(\d{2}[/°000']?\d{2}[/°000']?\d{2})"  
    time_pattern = r"TIME\s*(\d{2}:\d{2})"  
    withdrawal_pattern = r"(WITHDRAWAL)"  
    from_account_pattern = r"FROM\s*ACCOUNT\s*([A-Z0-9]+)"  
    amount_pattern = r"AMCUNT\s*([\d,]+\.\d{2})"  
    fee_amount_pattern = r"FEE\s*AMCUNT\s*([\d,]+\.\d{2})"  
    ac_balance_pattern = r"AC\s*BALANCE\s*([\d,]+\.\d{2})"  
  
    for i, line in enumerate(lines):  
        line_clean = line.replace(' ', ' ').replace('o', '0').replace('0', '0').strip()  
  
        # extract DATE  
        date_match = re.search(date_pattern, line_clean, re.IGNORECASE)  
        if date_match and extracted_data["date"] is None:  
            extracted_data["date"] = date_match.group(1).replace("/", "")  
  
        # extract TIME  
        time_match = re.search(time_pattern, line_clean, re.IGNORECASE)  
        if time_match and extracted_data["time"] is None:  
            extracted_data["time"] = time_match.group(1) + ":00"  
  
        # extract type of transaction (WITHDRAWAL)  
        withdrawal_match = re.search(withdrawal_pattern, line_clean, re.IGNORECASE)  
        if withdrawal_match and extracted_data["transaction_type"] is None:  
            extracted_data["transaction_type"] = withdrawal_match.group(1).upper()  
  
        # extract FROM ACCOUNT  
        from_account_match = re.search(from_account_pattern, line_clean, re.IGNORECASE)  
        if from_account_match and extracted_data["from_account"] is None:  
            extracted_data["from_account"] = from_account_match.group(1)  
  
        # extract AMOUNT (Withdrawal Amount)  
        amount_match = re.search(amount_pattern, line_clean, re.IGNORECASE)  
        if amount_match and extracted_data["withdrawal_amount"] is None:  
            extracted_data["withdrawal_amount"] = amount_match.group(1)  
  
        # extract FEE AMOUNT  
        fee_amount_match = re.search(fee_amount_pattern, line_clean, re.IGNORECASE)  
        if fee_amount_match and extracted_data["fee_amount"] is None:  
            extracted_data["fee_amount"] = fee_amount_match.group(1)  
  
        # extract A/C BALANCE  
        ac_balance_match = re.search(ac_balance_pattern, line_clean, re.IGNORECASE)  
        if ac_balance_match and extracted_data["account_balance"] is None:  
            extracted_data["account_balance"] = ac_balance_match.group(1)  
  
    return extracted_data
```

ภาพการโปรแกรม Text Classification ของธนาคารกสิกร



ภาพที่ใช้ในการทำ OCR ของธนาคารกสิกรไทย

```

0 pyesseract.image_to_json(pyimg)
1 {
2   "lines": [
3     "จ ATEA eae",
4     "",
5     "aut 152 ee eects",
6     "",
7     "กรลา Ber",
8     "",
9     "บันทึกการ เอนอ",
10    "",
11    "TRANSACTION RECORD. vs:",
12    "",
13    "สาขาเมืองทองธานี สาขาที่ ลำดับ, ง",
14    "และ Bonk เทไซ ae",
15    ],
16    "KEMK 5162915 4048 au oo\"",
17    "",
18    "DATE 04.04/25 TIME 14:02 en Pad",
19    "",
20    "TI THDRARAL Ss",
21    "",
22    "FROM ACCOUNT REX Z09TOR ae",
23    "",
24    "เอน",
25    "",
26    "AMOUNT 1,000. 00 a",
27    "",
28    "FEE AMOUNT 0.00 =",
29    "",
30    "AC BALANCE 4,077.82 ee",
31    "",
32    "ald แงมมม0677016161 =",
33    "ee"
34  ]
35 }

```

ผลลัพธ์ของการทำ OCR ของธนาคารกสิกรไทย

```

*****
{
  "date": null,
  "time": "14:02:00",
  "transaction_type": null,
  "from_account": null,
  "withdrawal_amount": null,
  "fee_amount": null,
  "account_balance": "4,077.82"
}
*****

```

ผลลัพธ์ของการทำ Text Classification ของธนาคารกสิกรไทย

การทำ Text Classification ของธนาคารกรุงไทย ข้อมูลที่ต้องการสกัดจากใบเสร็จธนาคารกรุงไทย มีดังนี้ วันที่, เวลา, ประเภทการทำธุรกรรม, จำนวนเงิน, และเลขบัญชี

```
def krungthai_extracted(lines):  
    extracted_data = {  
        "date": None,  
        "time": None,  
        "transaction_type": None,  
        "deposit_amount": None,  
        "ac_name": None  
    }  
  
    date_pattern = r"DATE\s*([\d]{2}/[\d]{2}/[\d]{2})"  
    time_pattern = r"TIME\s*([\d]{2}:[\d]{2})"  
    deposit_pattern = r"(AUTO\s*DEP)"  
    amount_pattern = r"จำนวนเงิน\s*([\d,]+\.\d{2})\s*BAHT"  
    ac_name_pattern = r"To A/C Name\s*: \s*(.+)"  
  
    for line in lines:  
        line_clean = line.strip()  
  
        # extract DATE  
        date_match = re.search(date_pattern, line_clean, re.IGNORECASE)  
        if date_match and extracted_data["date"] is None:  
            extracted_data["date"] = date_match.group(1)  
  
        # extract TIME  
        time_match = re.search(time_pattern, line_clean, re.IGNORECASE)  
        if time_match and extracted_data["time"] is None:  
            extracted_data["time"] = time_match.group(1) + ":00"  
  
        # extract type of transaction (AUTO DEP)  
        deposit_match = re.search(deposit_pattern, line_clean, re.IGNORECASE)  
        if deposit_match and extracted_data["transaction_type"] is None:  
            extracted_data["transaction_type"] = deposit_match.group(1)  
  
        # extract deposit amount  
        amount_match = re.search(amount_pattern, line_clean, re.IGNORECASE)  
        if amount_match and extracted_data["deposit_amount"] is None:  
            extracted_data["deposit_amount"] = amount_match.group(1)  
  
        # extract A/C Name  
        ac_name_match = re.search(ac_name_pattern, line_clean)  
        if ac_name_match and extracted_data["ac_name"] is None:  
            extracted_data["ac_name"] = ac_name_match.group(1).strip()  
  
    return extracted_data
```

ภาพการโปรแกรม Text Classification ของธนาคารกรุงไทย



ภาพที่ใช้ในการทำ OCR ของธนาคารกรุงไทย

```

1  {} pyesseract.json > [ ] lines > 12
2  {
3  |   "lines": [
4  |       "ทดสอบนิดเน่ ไม",
5  |       "}",
6  |       "yee, Krungthai",
7  |       "PY กรุงไทย",
8  |       ".ใบบันทึกรายการ",
9  |       "TRANSACTION RECORD",
10 |       "จงจ่ากใส่ไนแซ",
11 |       "อแล fave (ยอสบอ inner aon",
12 |       "0704/60 16:24 3008 awa",
13 |       "ททมิญชีเลขี่ เ่งมิญชีเลขี่ ประเทศรายการ",
14 |       "trom เจ Fransumtee",
15 |       "ร AUTO DFP",
16 |       "To AAG ผลพย :พาครกธิด ทารโรจธี่",
17 |       "จำนวยเจีย จำแนมมบัต",
18 |       "4คละดะซึ่งคะ Denomination",
19 |       "วอ,๑๐๐ ๐ 1๐๐๐%2๐ 30016 40๐๓",
20 |       "สำกบที่ ค้า๕55เพียน",
21 |       "No. te",
22 |       "เก ม",
23 |       "อแม7๒พีสินิตรศรมไทย.",
24 |       "เศศศอลบศ",
25 |   ]
26 }

```

ผลลัพธ์ของการทำ OCR ของธนาคารกรุงไทย

```

*****
{
  "date": null,
  "time": null,
  "transaction_type": null,
  "deposit_amount": null,
  "ac_name": null
}
*****

```

ผลลัพธ์ของการทำ Text Classification ของธนาคารกรุงไทย

ผลลัพธ์ของการทำ OCR ของธนาคารไทยพาณิชย์

ข้อมูลที่ต้องการสกัดจากใบเสร็จธนาคารไทยพาณิชย์จะแสดงผลข้อความจากที่สกัดได้จาก OCR เลย



ภาพที่ใช้ในการทำ OCR ของธนาคารไทยพาณิชย์

```
*****
[
  "\u0e40\u0e28\u0e25\u0e11\u0e4c\u0e23\u0e42\u0e23\u0e07\u0e23\u0e25\u0e13\u0e32",
  "\u0e1c\u0e1d",
  "\u0e41\u0e23\u0e30\u0e01\u0e23\u0e1e\u0e25\u0e01\u0e23",
  "\u0e40\u0e27\u0e2d ha",
  "oe 900.00",
  "moat 0.90",
  "|",
  "0",
  "2",
  "on"
]
*****
```

ผลลัพธ์ของการทำ OCR ของธนาคารไทยพาณิชย์

### 3.2.2 โครงสร้างภาพรวมของระบบ

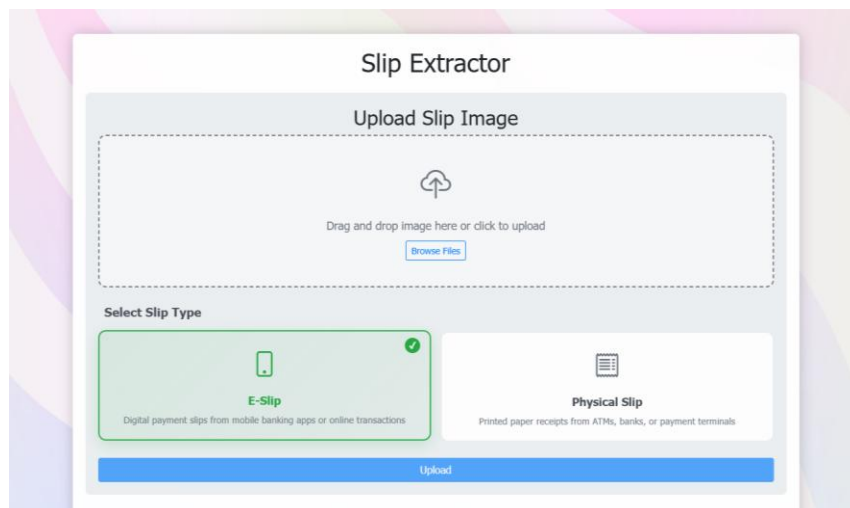
เมื่อระบบได้รับภาพจากผู้ใช้งานระบบแล้ว จะนำเข้าโมเดล YOLO และทำการ Preprocess ก่อนโดยการทำให้เป็น grayscale, gaussian blur, adaptive thresholding, และ binary inverse ตามลำดับ จากนั้นจึงนำภาพไปทำการ Perspective Transformation จากนั้นจึงนำเข้าไปสกัดตัวอักษรใน Pytesseract และใช้ Regular Expression สำหรับแยกข้อมูลที่ต้องการออกมา

### 3.3 การพัฒนาในรูปแบบ Web application

#### 3.3.1 โครงสร้างภาพรวมของระบบ

Web application ที่พัฒนาด้วย Flask framework โดยการประมวลผลต่างๆ ของรูปภาพจะประมวลผลในฝั่ง server side ทั้งหมด เป็นการนำ process ต่างๆมาเขียนเป็นฟังก์ชันให้ user เรียกใช้ผ่าน Rest API ได้ โดย Web application มีฟีเจอร์ดังนี้

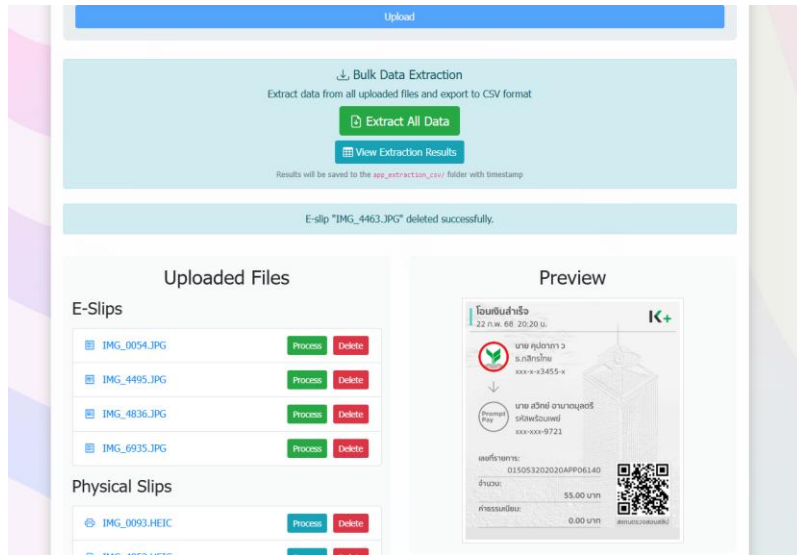
1. อัปโหลดรูปภาพใบเสร็จ และเลือกชนิดของใบเสร็จที่จะอัปโหลด



รูปภาพแสดง UI การอัปโหลดรูปภาพใบเสร็จ

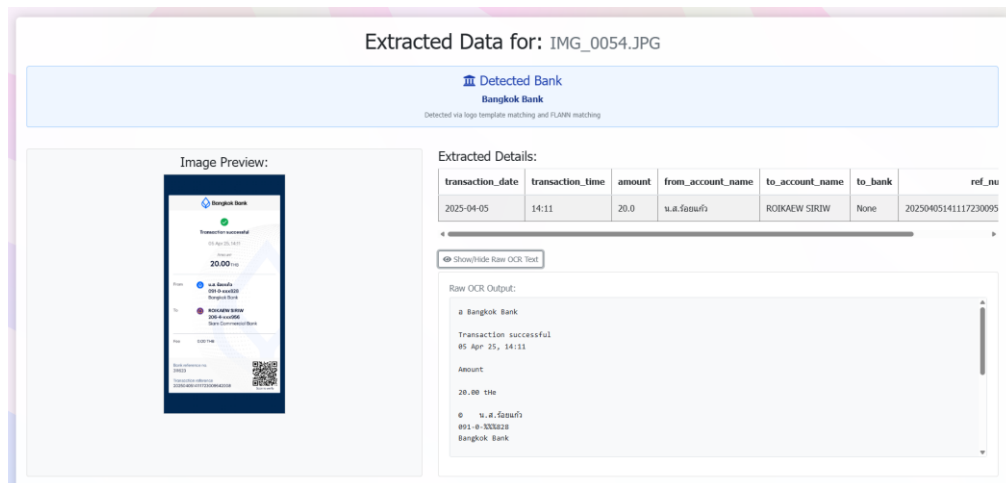
2. ดูรายการรูปภาพทั้งหมดที่ user อัปโหลดไว้





รูปภาพแสดง UI การดูรายการใบเสร็จ

3. สกัดข้อมูลออกมาจากรูปภาพในรูปแบบตาราง และดูผลลัพธ์ดิบของการ OCR ได้



รูปภาพแสดง UI การดูผลลัพธ์การประมวลผลของแต่ละใบเสร็จ

4. สามารถสกัดข้อมูลทั้งหมดที่ user อัปโหลด แสดงผลออกมาเป็นตาราง และบันทึกเป็นไฟล์ csv ได้

Extraction Results

← Back to Upload

Extraction Summaries

extraction\_summary\_20250531\_190005.csv

Created: 2025-05-31 19:00:25

Download CSV

processing_timestamp	total_e_slips_processed	total_physical_slips_processed	total_files_processed	total_failed_files	failed_files_list
2025-05-31 19:00:25	4	3	7	0	NaN

E-Slip Extraction Results

e\_slip\_extraction\_20250531\_190005.csv

4 records | Created: 2025-05-31 19:00:25

View DataDownload CSV

transaction_date	transaction_time	amount	from_account_name	to_account_name	to_bank	ref_number	from_bank	filename	slip_type	processing_timestamp	detected_bank
2025-04-05	14:11	20.0	บ.ส.ธนบุรี	ROKAEW SBRW	NaN	202504051411723009542008	bangkok	IMG_0054.JPG	e-slip	2025-05-31 19:00:06	bangkok
2025-02-22	20:20	55.0	นาย อดิชากร	ร.ภักดีไทย	NaN	015053202020APP06140	kbank	IMG_4495.JPG	e-slip	2025-05-31 19:00:08	kbank
NaN	NaN	120.0	Oe Krunattai	NaN	NaN	NaN	krungthai	IMG_4836.JPG	e-slip	2025-05-31 19:00:12	krungthai
2024-08-15	08:04	100.0	น. วัฒนศิริวัฒน์	NaN	NaN	2024081510059PKCumtUpHt	scb	IMG_6935.JPG	e-slip	2025-05-31 19:00:17	scb

Physical Slip Extraction Results

physical\_slip\_extraction\_20250531\_190005.csv

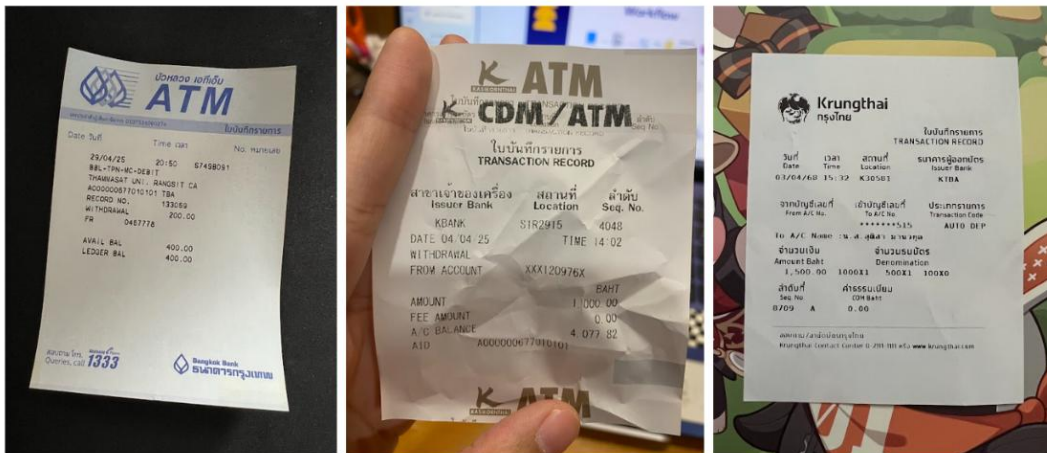
View DataDownload CSV

รูปภาพแสดง UI การดูผลลัพธ์การประมวลผลใบเสร็จทั้งหมด

## 3.4 เครื่องมือที่ใช้ในการพัฒนา

### 3.4.1 ภาพนำเข้า

ภาพนำเข้าเป็นภาพถ่ายจากโทรศัพท์มือถือแสดงรูปใบเสร็จจากการทำธุรกรรมจากตู้ ATM



ภาพตัวอย่างภาพใบเสร็จจากตู้ ATM

### 3.4.2 โปรแกรม Visual Studio Code (VS Code)

Visual Studio Code เป็นโปรแกรมที่ใช้สำหรับการพัฒนาระบบในส่วนใบเสร็จการทำธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ โปรแกรมนี้มีประสิทธิภาพสูง และมีจุดเด่นหลักในเรื่องของส่วนต่อขยายหรือ Extension ที่ทำให้ผู้พัฒนาสามารถเขียนโปรแกรมได้หลายภาษาจากการติดตั้งเพียงครั้งเดียว



ภาพโปรแกรม Visual Studio Code

อ้างอิง: <https://webdodee.com/what-is-visual-studio-code-and-how-to-use/>

### 3.4.3 Jupyter Notebook

Jupyter Notebook เป็นโปรแกรมที่ทำให้ผู้พัฒนาสามารถเขียนโปรแกรมบนเว็บไซต์ได้ โดย Jupyter Notebook ก็เป็นส่วนหนึ่งของ Extension บน VS Code ด้วยเช่นกัน โดยโปรแกรมนี้นี้มีลักษณะที่ใน 1 ไฟล์ สามารถสร้างช่องย่อย ๆ ได้หลายช่องเรียกว่า cell และการ run แต่ละ cell สามารถแทน run python file ปกติได้เลย ผลลัพธ์ที่ออกมาจะอยู่ใต้ cell ดังกล่าว ทำให้โปรแกรมในลักษณะนี้เหมาะกับการลองผิดลองถูก



ภาพโปรแกรม Jupyter Notebook

อ้างอิง: [https://en.wikipedia.org/wiki/Project\\_Jupyter](https://en.wikipedia.org/wiki/Project_Jupyter)

### 3.4.4 YOLO V8

You Only Look Once หรือ YOLO เวอร์ชัน 8 เป็น Object Detection Model ที่มีความสามารถในการตรวจจับวัตถุที่รวดเร็วและแม่นยำ อีกทั้งยังสามารถตรวจจับได้แบบ Realtime นอกจากนี้ YOLO ยังมีความสามารถในการตรวจจับวัตถุที่ซ้อนทับกันบนภาพได้อีกด้วย จึงเหมาะกับงานที่ต้องการความเร็วในการตรวจจับวัตถุ เช่น ตรวจจับวัตถุในวิดีโอ เป็นต้น



ภาพ Logo โปรแกรม YOLO

อ้างอิง: <https://grassrootengineer.medium.com/yolo>

### 3.4.5 Flask

Flask (ฟลัสก์) คือ ไมโครเว็บเฟรมเวิร์ก (micro web framework) สำหรับภาษาโปรแกรม Python เป็นเครื่องมือที่ช่วยให้นักพัฒนาสามารถสร้างเว็บแอปพลิเคชันและ API (Application Programming Interfaces) ได้อย่างรวดเร็วและง่ายดาย



ภาพ Logo Flask web framework

อ้างอิง: [https://en.wikipedia.org/wiki/Flask\\_%28web\\_framework%29](https://en.wikipedia.org/wiki/Flask_%28web_framework%29)

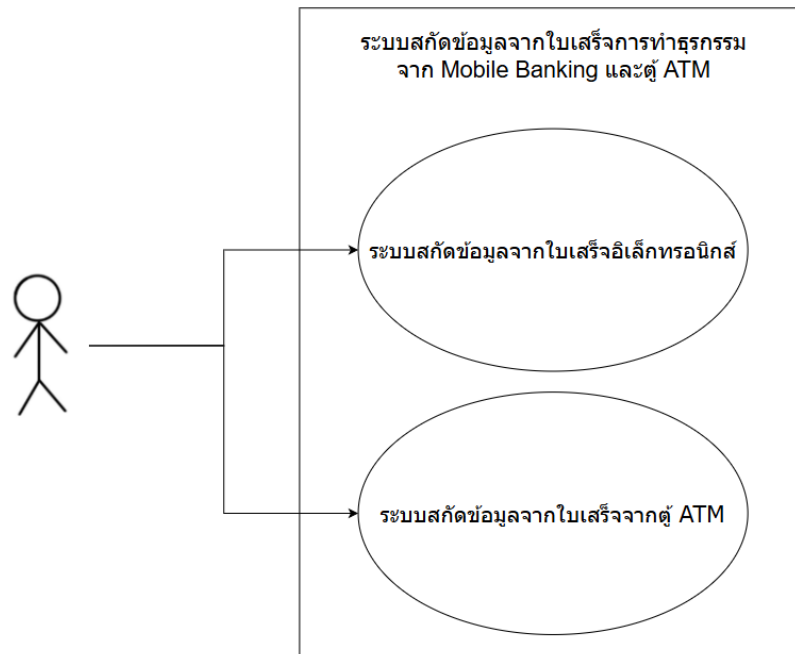
### 3.4.6 Tesseract OCR

Python Tesseract OCR คือชุดเครื่องมือที่ใช้ในการประมวลผล Optical Character Recognition (OCR) ในภาษา Python โดยเป็น Wrapper (ส่วนห่อหุ้ม) สำหรับ Google's Tesseract-OCR Engine

โดยมี Pytesseract (Python-tesseract) คือไลบรารี Python ที่ทำหน้าที่เป็นตัวกลางในการเชื่อมต่อกับ Tesseract OCR Engine แทนที่เราจะต้องเรียกใช้ Tesseract ผ่าน Command Line โดยตรง Pytesseract ช่วยให้เราสามารถเรียกใช้ฟังก์ชันต่างๆ ของ Tesseract ได้อย่างง่ายดายภายในโค้ด Python

## 3.5 การวิเคราะห์และขอบเขตความต้องการระบบ

### 3.5.1 Use case diagram



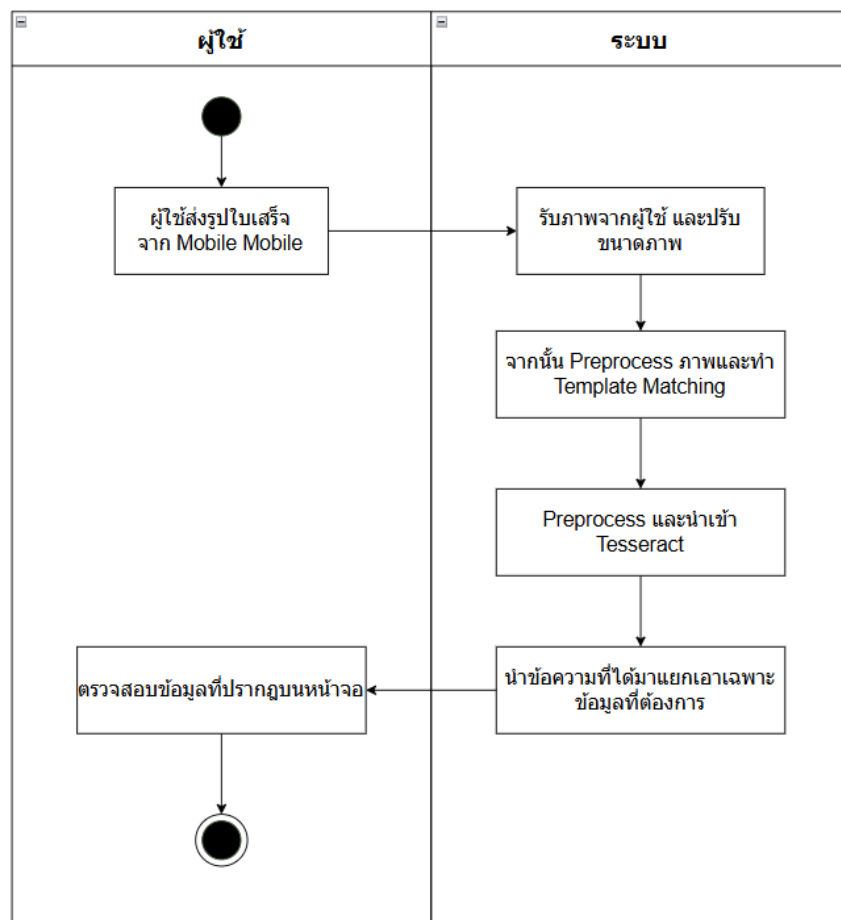
ภาพ Use Case Diagram

### 3.5.2 Use case description

ชื่อ Use case (Use case Name)	ส่งภาพใบเสร็จเข้าสู่ระบบ
ผู้ใช้ (Actor)	ผู้ที่ต้องการทำรายรับรายจ่าย หรือร้านค้า SME ที่ต้องการทำบัญชีการค้าขาย
คำอธิบาย (Description)	ผู้ใช้นำภาพใบเสร็จที่ต้องการสกัดข้อมูลเข้าสู่ระบบ
เงื่อนไขก่อนหน้า (Precondition)	ภาพใบเสร็จที่ตรงตามข้อกำหนดที่ถูกระบุไว้
เงื่อนไขภายหลัง (Post condition)	ผู้ใช้ได้รับข้อมูลบนใบเสร็จเป็น text ตามที่ต้องการ
กระแสหลัก (Basic flow)	<ol style="list-style-type: none"> <li>1. Use case เริ่มก็ต่อเมื่อ user มีรูปภาพใบเสร็จ</li> <li>2. User เลือกชนิดของภาพก่อนอัปโหลดเข้าไปในระบบ</li> <li>3. User อัปโหลดรูปภาพไปในระบบ <ol style="list-style-type: none"> <li>a. รูปภาพใบเสร็จจาก ATM <ol style="list-style-type: none"> <li>i. ระบบเลือกใช้ฟังก์ชันประมวลผลสำหรับใบเสร็จจาก ATM</li> </ol> </li> </ol> </li> </ol>

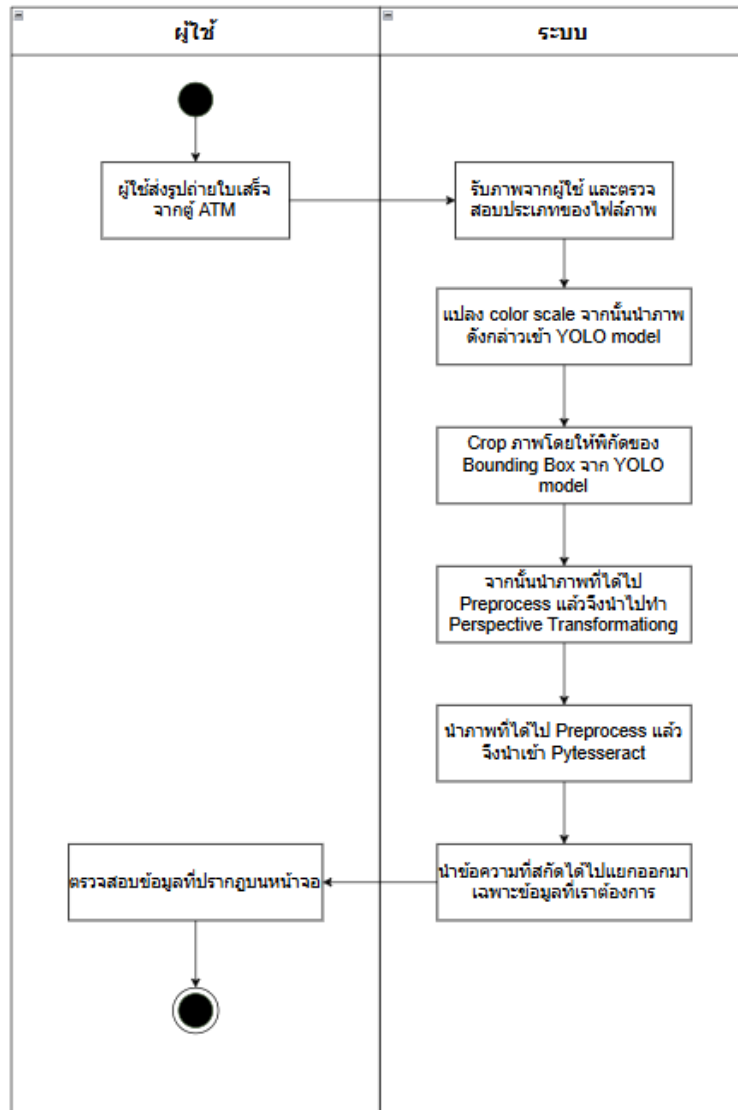
	b. รูปภาพสลิปจาก Mobile Banking i. ระบบเลือกใช้ฟังก์ชันประมวลผลสำหรับ ใบเสร็จจาก Mobile Bankin 4. ระบบจำแนกธนาคารและสกัดข้อมูลออกมาเป็นตาราง
กระแสรอง (Alternative flow)	ระบบไม่รองรับรูปภาพที่ User อัปโหลด

### 3.5.3 Activity Diagram ของระบบสกัดข้อมูลสำหรับใบเสร็จอิเล็กทรอนิกส์การทำธุรกรรมจากแอปพลิเคชันธนาคาร (Mobile Banking)



ภาพ Activity Diagram ของระบบสกัดข้อมูลจากใบเสร็จการทำธุรกรรมจาก Mobile Banking

### 3.5.4 Activity Diagram ของระบบสกัดข้อมูลจากใบเสร็จการทำธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ (ATM)



ภาพ Activity Diagram ของระบบสกัดข้อมูลจากใบเสร็จการทำธุรกรรมจากเครื่องรับจ่ายเงินอัตโนมัติ

## บทที่ 4

### การทดลอง

ในขั้นตอนนี้คือการทดลองขั้นตอนการสกัดข้อความจากใบเสร็จจาก Mobile Banking และใบเสร็จจากตู้ ATM เพื่อวัดประสิทธิภาพความถูกต้องของระบบทั้งหมด ซึ่งประกอบไปด้วย 2 ระบบย่อย คือ ระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking และระบบสกัดข้อความจากใบเสร็จจากตู้ ATM โดยแต่ละระบบย่อยจะถูกแบ่งการทดลองออกเป็น 2 ส่วน ส่วนแรก คือ การทดลองเพื่อวัดประสิทธิภาพการตรวจจับธนาคารเจ้าของใบเสร็จ และส่วนที่สอง คือ การทดลองเพื่อวัดประสิทธิภาพการสกัดข้อความ ความถูกต้องของผลลัพธ์วัดจากร้อยละของความถูกต้องของภาพที่ทดสอบทั้งหมด โดยมีตัวแปรในการควบคุมการทดลอง ดังนี้

**ตัวแปรต้น :** รูปภาพใบเสร็จจาก Mobile Banking หรือใบเสร็จจากตู้ ATM จากแต่ละธนาคาร

**ตัวแปรตาม :** ร้อยละความแม่นยำของการตรวจจับธนาคารเจ้าของใบเสร็จ และร้อยละของภาพที่สกัดข้อความที่ต้องการออกมาได้อย่างถูกต้อง

**สมมติฐานในการทดลอง คือ**

1. สำหรับระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking
  - สมมติฐานที่ 1 : อัตราการตรวจจับธนาคารเจ้าของใบเสร็จได้ถูกต้องมากกว่าร้อยละ 90
  - สมมติฐานที่ 2 : ประสิทธิภาพของการสกัดข้อความถูกต้องมากกว่าร้อยละ 60
  - สมมติฐานที่ 3 : ระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking ใช้เวลาในการประมวลแต่ละครั้งไม่เกิน 10 วินาที
2. สำหรับระบบสกัดข้อความจากใบเสร็จจาก ATM
  - สมมติฐานที่ 1 : อัตราการตรวจจับธนาคารเจ้าของใบเสร็จได้ถูกต้องมากกว่าร้อยละ 90
  - สมมติฐานที่ 2 : ประสิทธิภาพของการสกัดข้อความถูกต้องมากกว่าร้อยละ 60
  - สมมติฐานที่ 3 : ระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking ใช้เวลาในการประมวลแต่ละครั้งไม่เกิน 10 วินาที



#### 4.1 เครื่องมือในการทดลอง

##### 1. ภาพใบเสร็จจากตู้ Mobile Banking ของแต่ละธนาคาร

ภาพใบเสร็จ Mobile Banking 4 ธนาคาร ได้แก่ ธนาคารกรุงเทพ ธนาคารกสิกรไทย ธนาคารกรุงไทย และธนาคารไทยพาณิชย์ ที่บันทึกไว้หลังจากทำธุรกรรมเสร็จสิ้น จำนวนทั้งหมด 129 ภาพ

##### 2. ภาพใบเสร็จจากตู้ ATM ของแต่ละธนาคาร

ภาพใบเสร็จจากตู้ของ 4 ธนาคาร ได้แก่ ธนาคารกรุงเทพ ธนาคารกสิกรไทย ธนาคารกรุงไทย และธนาคารไทยพาณิชย์ จำนวน 30 ภาพ ซึ่งเป็นภาพที่มีพื้นหลังสีพื้นและพื้นหลังลาย ระยะการถ่ายแบบใกล้และไกล ใบเสร็จยับและใบเสร็จเรียบ ภาพเหล่านี้ถูกถ่ายในแสงที่ต่างกัน

#### 4.2 การทดลองของระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking

##### 4.2.1 เครื่องมือในการทดลอง

##### 4.2.2 การทดลองเพื่อวัดประสิทธิภาพการตรวจจับธนาคารเจ้าของใบเสร็จ

##### 4.2.2.1 ขั้นตอนในการทดลอง

การทดลองนี้ นำภาพถ่ายจาก Mobile Banking จาก 4 ธนาคาร ได้แก่ ธนาคารกรุงเทพ ธนาคารกสิกรไทย ธนาคารกรุงไทย และธนาคารไทยพาณิชย์ จำนวนทั้งหมด 129 ภาพ เป็น

- ธนาคารกสิกรไทย 58 ภาพ
- ธนาคารกรุงเทพ 12 ภาพ
- ธนาคารกรุงไทย 18 ภาพ
- ธนาคารไทยพาณิชย์ 41 ภาพ

นำทั้งหมดมาประมวลผลและเก็บข้อมูลในรูปแบบไฟล์ .csv

##### 4.2.2.2 ผลการทดลอง

ตารางแสดงจำนวนผลลัพธ์การตรวจจับธนาคารเจ้าของใบเสร็จจาก Mobile Banking

Bank	Total Records	Valid Classification results	Accuracy
------	---------------	------------------------------	----------

Bankok	12	12	100%
KBank	58	58	100%
Krungthai	18	18	100%
SCB	41	41	100%

ระบบสามารถตรวจจับธนาคารจากใบเสร็จ Mobile Banking ทั้งหมด 4 ธนาคารได้ 100%

#### 4.2.3 การทดลองเพื่อวัดประสิทธิภาพการสกัดข้อความ

##### 4.2.3.1 ขั้นตอนในการทดลอง

การทดลองนี้นำภาพถ่ายจาก Mobile Banking จาก 4 ธนาคาร ได้แก่ ธนาคารกรุงเทพ ธนาคารกสิกรไทย ธนาคารกรุงไทย และธนาคารไทยพาณิชย์ จำนวนทั้งหมด 129 ภาพ เป็น

- ธนาคารกสิกรไทย 58 ภาพ
- ธนาคารกรุงเทพ 12 ภาพ
- ธนาคารกรุงไทย 18 ภาพ
- ธนาคารไทยพาณิชย์ 41 ภาพ

นำทั้งหมดมาประมวลผลและเก็บข้อมูลในรูปแบบไฟล์ .csv

##### 4.2.3.2 ผลการทดลอง

ตารางแสดงผลลัพธ์การสกัดข้อมูลใบเสร็จจาก Mobile Banking ของแต่ละ Column

Total Missing Data คือข้อมูลที่ระบบไม่สามารถสกัดออกมาได้ของแต่ละ Column

Column	Total Records	Total Missing Data	Success Rate
Transaction_date	129	96	25.6%
Transaction_time	129	96	25.6%
amount	129	9	93.0%
from_account_name	129	6	95.3%
to_account_name	129	22	82.9%
to_bank	129	108	16.3%
ref_number	129	18	86.0%

ระบบสามารถสกัดข้อมูล amount, from\_account\_name, to\_account\_name, และ ref\_number ได้ดี

ตารางแสดงผลลัพธ์การสกัดข้อมูลใบเสร็จจาก Mobile Banking ในทุก Column ของแต่ละธนาคาร

Bank	Total Records	Overall Columns Success Rate
Bankok	12	79.8%
KBank	58	60.6%
Krungthai	18	50.0%
SCB	41	59.0%

ระบบสามารถสกัดข้อมูลธนาคารกรุงเทพได้ดีที่สุด

#### 4.3 การทดลองของระบบสกัดข้อความจากใบเสร็จจากตู้ ATM

##### 4.3.1 เครื่องมือในการทดลอง

##### 4.3.2 การทดลองเพื่อวัดประสิทธิภาพการตรวจจับธนาคารเจ้าของใบเสร็จจากตู้ ATM

##### 4.3.2.1 ขั้นตอนในการทดลอง

การทดลองนี้นำภาพถ่ายใบเสร็จจากตู้ ATM จาก 4 ธนาคาร ได้แก่ ธนาคารกรุงเทพ ธนาคารกสิกรไทย ธนาคารกรุงไทย และธนาคารไทยพาณิชย์ ซึ่งเป็นภาพที่มีพื้นหลังสีพื้นและพื้นหลังลาย ระยะการถ่ายแบบใกล้และไกล ใบเสร็จยับและใบเสร็จเรียบ ภาพเหล่านี้ถูกถ่ายในแสงที่ต่างกัน ทั้งหมดจำนวน 30 ภาพ ประกอบด้วยภาพถ่ายใบเสร็จของธนาคารกรุงเทพจำนวน 6 ภาพ, ภาพถ่ายใบเสร็จของธนาคารกสิกรไทยจำนวน 11 ภาพ, ภาพถ่ายใบเสร็จของธนาคารกรุงไทยจำนวน 8 ภาพ, และภาพถ่ายใบเสร็จของธนาคารไทยพาณิชย์จำนวน 5 ภาพ การทดลองนี้จะนำภาพไปตรวจจับธนาคารเจ้าของใบเสร็จ โดยมีขั้นตอนดังนี้

1. เตรียมภาพถ่ายใบเสร็จจากตู้ ATM
2. นำภาพถ่ายที่เตรียมไว้ไปตรวจจับธนาคารเจ้าของใบเสร็จ
3. จัดบันทึกผลลัพธ์ที่ได้

##### 4.3.2.2 ผลการทดลอง

Confusion Metrix

ตารางแสดงจำนวนผลลัพธ์การตรวจจับธนาคารเจ้าของใบเสร็จ

ภาพนำเข้า	ผลลัพธ์ของระบบ					
		กรุงเทพ	กสิกรไทย	กรุงไทย	ไทยพาณิชย์	ไม่ระบุผลลัพธ์
กรุงเทพ		5	-	-	1	-

	กสิกรไทย	-	10	-	-	1
	กรุงไทย	-	1	7	-	-
	ไทยพาณิชย์	-	-	-	5	-

Efficientcy = จำนวนภาพที่หายถูก / จำนวนภาพทั้งหมด = 27 / 30 = 90%

#### 4.3.3 การทดลองเพื่อวัดประสิทธิภาพการสกัดข้อความ

##### 4.3.3.1 ขั้นตอนในการทดลอง

การทดลองนี้จะนำภาพที่ถูกตรวจจับธนาคารเจ้าของใบเสร็จมาแล้ว นำไปทำการสกัดข้อความ โดยเป็นภาพที่มีพื้นหลังสีพื้นและพื้นหลังลาย ระยะการถ่ายแบบใกล้และไกล ใบเสร็จยับและใบเสร็จเรียบ ภาพเหล่านี้ถูกถ่ายในแสงที่ต่างกัน โดยแต่ละธนาคารจะมีข้อมูลที่สกัดออกมาได้ต่างกันออกไป โดยมีขั้นตอนในการทดลองดังนี้

1. เตรียมภาพถ่ายจากขั้นตอนการตรวจจับธนาคารเจ้าของบัตร
2. นำภาพเข้าไปสกัดข้อความ
3. จัดบันทึกผลลัพธ์

##### 4.3.3.2 ผลการทดลอง

##### Confusion Metrix

##### ตารางแสดงจำนวนผลลัพธ์การสกัดข้อความจากใบเสร็จจากตู้ ATM

หมายเหตุ (A/B) แทนภาพที่นำมาทดสอบแต่ละภาพ โดย A คือ จำนวนข้อมูลที่สามารถสกัดได้ของแต่ละธนาคาร และ B คือ จำนวนข้อมูลทั้งหมดที่เราต้องการของแต่ละธนาคาร

	จำนวนภาพที่นำเข้าระบบ	จำนวนภาพที่ถ่ายถูก
กรุงเทพ	5	(0/5), (error), (error), (error), (error)
กสิกรไทย	10	(2/7), (0/7), (0/7), (0/7), (error), (0/7), (0/7), (0/7), (error), (0/7)
กรุงไทย	7	(0/5), (0/5), (0/5), (0/5), (0/5), (0/5), (0/5)
ไทยพาณิชย์	5	-

Efficiency ใกล้ศูนย์

## บทที่ 5

### สรุป อภิปรายผล และข้อเสนอแนะ

#### 5.1 สรุปผลการดำเนินงาน

สำหรับส่วนระบบสกัดข้อความจากใบเสร็จจากตู้ ATM จะประกอบด้วย 2 ระบบย่อย คือ การตรวจจับธนาคารเจ้าของใบเสร็จ และการสกัดข้อความจากใบเสร็จ โดยก่อนจะเริ่มขั้นตอนการตรวจจับธนาคารเจ้าของใบเสร็จ เราต้องเริ่มจากการเตรียมข้อมูลโดยการ label ข้อมูลเพื่อนำไป train YOLO model หลังจากนั้นก็พัฒนาระบบต่อ โดยการนำผลลัพธ์ที่ได้จาก YOLO model นั้นก็คือ พิกัด Bounding Box ไปเป็นพิกัดของการ crop รูปให้ได้ Region of Interest แล้วนำภาพที่ได้ไป Preprocess เช่น grayscale, adaptive thresholding, และ binary inverse จากนั้นนำภาพที่ผ่านการ Preprocess แล้วไปทำ OCR โดยในขบวนการทำการเปรียบเทียบแล้วว่า การ Preprocess แบบไหนเหมาะกับโครงงานนี้มากที่สุด ระหว่าง grayscale, binary, และการทำงานร่วมกันของ grayscale, adaptive thresholding, และ binary inverse โดยเทคนิคที่เหมาะสมที่สุดคือ การใช้ร่วมกันของ 3 เทคนิค หลังจากนั้นนำข้อความที่สกัดได้ไปคัดแยกข้อมูลที่ต้องการของแต่ละธนาคารออกมาโดยใช้ Regular Expression

ส่วนของระบบสกัดข้อความจากใบเสร็จจาก Mobile banking โดยการเตรียม logo ของธนาคารทั้งสี่ และนำใบเสร็จและ logo ไปทำ grayscale, Otsu thresholding นำไปทำ Template Matching และ FLANN Matching เพื่อจำแนกประเภทธนาคารของใบเสร็จ แล้วนำไปทำ OCR โดยใช้ PyTesseract เพื่อสกัดข้อมูล ตัวอักษรออกมา แล้วแยกข้อมูลออกมาเป็นตารางด้วย Regular Expression พบว่า thresholding แบบ Otsu ดีที่สุดในการลบพื้นหลังของรูปภาพออก ส่วนการ Template Matching อย่างเดียวอาจไม่เพียงพอต่อการจำแนกประเภท เนื่องจากมีโอกาสที่พื้นหลังของ slip จะเปลี่ยนไปตามเทศกาล และส่งผลให้ทำ Template Matching ได้ยากในบางครั้งจึงใช้ FLANN Matching เข้ามาช่วยให้การจำแนกธนาคารดีขึ้น

#### 5.2 อภิปรายผล

จากการทดลองของระบบสกัดข้อความจากใบเสร็จจากตู้ ATM การ train YOLO model ผ่านไปได้ด้วยดีไม่มีปัญหาอะไร แต่ข้อควรระวังในการทำคือเรื่องของ color scale ถ้า color scale ผิดทำให้สีเพี้ยน ในบางกรณีสามารถทำให้ผลของการทำนายเจ้าของธนาคารเจ้าของใบเสร็จเพี้ยนไปได้ อย่างไรก็ตาม ประเด็นดังกล่าวไม่ใช่ประเด็นที่ใหญ่ที่สุดของงานในส่วนระบบสกัดข้อความจากใบเสร็จจากตู้ ATM

ส่วนที่ยากที่สุดคือ การสกัดข้อความจากใบเสร็จที่มีความหลากหลายของคุณภาพใบเสร็จ มีตั้งแต่ใบเสร็จเรียบวางตรง ไปจนถึงใบเสร็จยับมากวางเอียง ๆ อีกทั้งยังมีเงาที่เกิดขึ้นบนใบเสร็จตั้งแต่สีอ่อน ไปจนถึงสีเข้ม แต่ละปัญหาที่เกิดขึ้นใช้เทคนิคในการปรับปรุงภาพที่ต่างกันเช่นกัน ใบเสร็จเรียบ ๆ สามารถใช้ได้ทั้งเทคนิค grayscale, binary แต่ binary จะเหมาะที่สุดเพราะสร้าง contrast ได้มากกว่า ส่วนใบเสร็จมีรอยยับ จะเหมาะกับเทคนิค binary และใบเสร็จที่มีเงาจะเหมาะกับเทคนิค adaptive thresholding ซึ่งถ้าใช้เทคนิคนี้กับใบเสร็จยับแล้วจะเกิด noise จำนวนมาก และถ้านำเทคนิคอย่าง morphology เช่น closing มาใช้จะไม่เหมาะสม เพราะตัวหนังสือที่ต้องการสกัดก็จะหายไปด้วย โดยปัจจุบันทางผู้จัดทำขอยอมรับจากใจจริงว่า ยังไม่สามารถหาเทคนิคที่สามารถจัดการใบเสร็จที่มีทั้งความยับและเงาที่เกิดขึ้นบนใบเสร็จได้ จึงจำเป็นต้องเลือกทางเลือกใดทางเลือกหนึ่ง ซึ่งผู้จัดทำขอเลือกสนใจใบเสร็จเรียบที่มีปัญหาเงาตกกระทบบนใบเสร็จ จากทั้งหมดที่กล่าวมา คุณภาพข้อความที่ถูกสกัดออกมาจึงค่อนข้างต่ำ จึงไม่สามารถแยกข้อมูลที่ต้องการออกมาได้ โดยในอนาคตผู้จัดทำหวังว่าจะสามารถรับมือกับปัญหาลักษณะเช่นนี้ได้ดียิ่งขึ้น

จากการทดลองของระบบสกัดข้อความจากใบเสร็จจาก Mobile banking พบว่าระบบสามารถจำแนกประเภทของธนาคารได้ดีมาก แต่การสกัดข้อความออกมากจากรูปภาพด้วย OCR ยังได้ผลลัพธ์ที่ตัวอักษรผิดเพี้ยนอยู่บ้างทำให้ยากต่อการ เขียน Regular Expression เพื่อจำแนกข้อมูลออกมา ซึ่งอาจจะมาจากฟอนต์ของตัวอักษรในใบเสร็จของแต่ละธนาคาร ที่ Tesseract OCR ที่อาจจะไม่ได้ถูกฝึกมาให้ใช้กับฟอนต์เหล่านี้ได้ดีมากนัก และมีโอกาสที่จะได้ผลลัพธ์ดีขึ้นหรือแย่ลงขึ้นอยู่กับพื้นหลังที่เปลี่ยนไปของธนาคารนั้นตามเทศกาลด้วย

### 5.3 ข้อจำกัดของระบบ

ระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking

1. ระบบสามารถสกัดได้ 4 ธนาคารเท่านั้น คือ ธนาคารกสิกรไทย, กรุงไทย, ไทยพาณิชย์ และ กรุงเทพ
2. ใบเสร็จต้องมี logo ของธนาคารนั้นๆ และตรงกับที่เตรียมไว้ ถึงจะสามารถจำแนกธนาคารนั้นๆได้

ระบบสกัดข้อความจากใบเสร็จจากตู้ ATM

1. ภาพที่ใส่เข้าระบบต้องเป็นภาพที่ถ่ายจากโทรศัพท์มือถือเท่านั้น
2. ควรใช้ใบเสร็จที่เรียบ
3. ควรถ่ายภาพในที่ ๆ แสงพอ

4. ขณะถ่ายภาพใบเสร็จควรจัดวางใบเสร็จให้ตรงมากที่สุด

5. ไม่ควรจัดวางกล้องถ่ายใบเสร็จเกิน 15 เซนติเมตร

แต่เพื่อผลลัพธ์ที่ดีที่สุดควรถ่ายภาพใบเสร็จให้พอดีกับภาพมากที่สุด

#### 5.4 ข้อเสนอแนะ

ระบบสกัดข้อความจากใบเสร็จจาก Mobile Banking

1. การแยกข้อมูลตัวอักษรจากรูปภาพด้วย Tesseract OCR ยังไม่ค่อยดีมากนัก อาจใช้โมเดลที่มีขนาดใหญ่กว่าให้ผลลัพธ์ออกมาดียิ่งขึ้น

ระบบสกัดข้อความจากใบเสร็จจากตู้ ATM

1. ควรกำหนดขนาดโปรเจคให้รัดกุมและรอบคอบ
2. ควรคำนึงถึงความยากง่ายในการเก็บข้อมูล dataset เป็นสำคัญ เพราะงานของเราจะมีประสิทธิภาพหรือไม่ขึ้นอยู่กับสิ่งนี้
3. ก่อนที่จะฝึกโมเดลต่าง ๆ ควรวางแผนให้ดีกว่าก่อนว่า ผลลัพธ์ที่จะได้จากการฝึกโมเดลที่จะเอาไปใช้ทำอะไร ยกตัวอย่างเช่น การฝึกโมเดลมาใช้ในโปรเจคนี้นี้ ใช้การ Label ก่อนการนำเข้าโมเดลฝึกแบบ Bounding box จึงได้ผลลัพธ์มาเป็นพิกัด Bounding Box อย่างไรก็ตาม เพิ่งมาทราบในภายหลังว่า อาจจะต้องมีการทำ Perspective Transformation ซึ่งถ้าได้ผลลัพธ์จากโมเดลเป็นพิกัดมุมม 4 มุมของใบเสร็จ จะทำให้ประหยัดเวลาได้มากขึ้น