# บทที่ 3

# ขั้นตอนการดำเนินงานวิจัย

ในบทนี้จะกล่าวถึงขั้นตอนในการดำเนินงานของโครงงานปริญญานิพนธ์ ซึ่งจะทำให้ทราบถึงการวิเคราะห์ระบบ กระบวนการประมวลของระบบ และขั้นตอนการทำงานของระบบเป็นอย่างไร โดยขั้นตอนในการดำเนินงานมีรายละเอียดดังนี้

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

## 3.1 ขั้นตอนการจัดเตรียมชุดข้อมูล

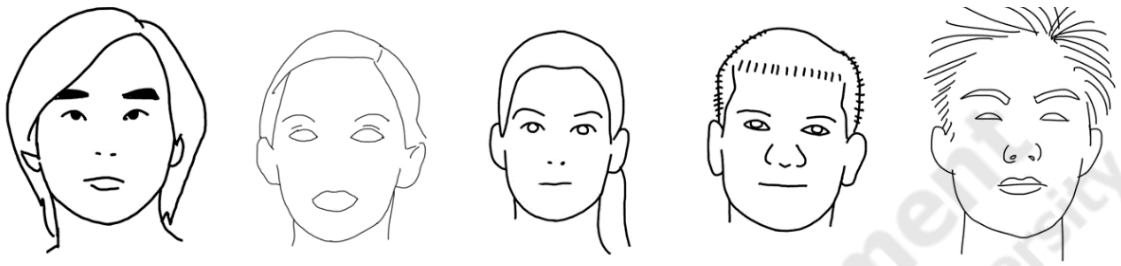
**ภาพประกอบที่ 3.1** ตัวอย่างภาพใบหน้าและภาพสเกตช์ใบหน้า
จาก CUFS และ Face Research Lab London

### 3.1.1 จัดเตรียมชุดข้อมูล

ระบบงานนี้ใช้ชุดข้อมูลจากฐานข้อมูล CUHK Face Sketch Database (CUFS) ที่มีจำนวนคู่ภาพใบหน้าและภาพสเกตช์ทั้งหมด 311 คู่ แต่เนื่องจากรูปแบบภาพสเกตช์ที่ต้องการคือภาพสเกตช์ที่มีไม่มีข้อมูลแสงเงา หรือเป็นภาพสเกตช์ในรูปแบบที่ยังไม่สมบูรณ์แบบ อีกทั้งยังมีจำนวนภาพสเกตช์ที่น้อย ไม่เพียงพอต่อการเรียนรู้และทดสอบโมเดล ผู้จัดทำจึงได้ทำการสร้างชุดข้อมูลขึ้นมาใหม่จากชุดข้อมูล CUFS และ Face Research Lab London ที่มีเพียงภาพใบหน้าอีกจำนวน 102 รูป

ผู้จัดทำได้ทำการสร้างภาพสเกตช์ขึ้นมาใหม่จากต้นแบบภาพใบหน้าที่ได้จากฐานข้อมูลที่กล่างมาข้างต้น โดยการสเกตช์ภาพด้วยเมาส์ปากกาและ iPad โปรแกรมที่ใช้ในการวาดภาพมี Photoshop, Procreate และ PaintToolSAI ได้จำนวน 900 รูป และมีอาสาสมัครอีก 3 คน ที่ช่วยในการสเกตช์

ภาพเพิ่มเติมอีก 300 รูป ได้ผลลัพธ์เป็นภาพสเกตช์ทั้งหมด 1200 รูปตัวอย่างภาพสเกตช์เป็นไปตามภาพประกอบที่ 3.2 ซึ่งแสดงให้เห็นถึงภาพสเกตช์ที่ไม่มีองค์ประกอบแสงเงา มีเพียงลายเส้นของเค้าโครงใบหน้าและองค์ประกอบเท่านั้น



**ภาพประกอบที่ 3.2** ตัวอย่างภาพสเกตช์ที่ต้องการในการเรียนรู้ของโมเดล

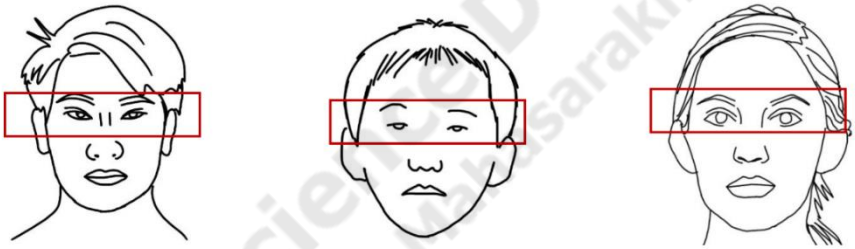
อีกทั้งผู้จัดทำยังได้ทำการสร้างภาพสเกตช์จากชุดข้อมูล CelebAMask-HQ ที่เป็นชุดข้อมูลภาพใบหน้าและหน้ากากแยกองค์ประกอบ จำนวนทั้งหมด 5,500 รูป มาใช้สร้างชุดข้อมูลภาพสเกตช์เพิ่มเติมอีก 5,500 รูป โดยการใช้วิธีการประยุกต์ใช้ Photocopy ภาพใบหน้า แล้วทำการสร้างภาพสเกตช์โดยใช้ระบบงาน Sketch Simplification แต่เนื่องจากไม่มีข้อมูลคำอธิบายองค์ประกอบใบหน้า จึงจำเป็นที่จะต้องสร้างชุดข้อมูลองค์ประกอบใบหน้าให้ครบทุกภาพที่สร้างขึ้นมาในภายหลัง

**ภาพประกอบที่ 3.3** ภาพสเกตช์ที่สร้างโดยใช้วิธี Photocopy ตามด้วย Sketch Simplification

ในส่วนของคำอธิบายภาพสเกตช์ของแต่ละภาพจะถูกกำหนดกำหนดขึ้นจากภาพต้นแบบ แล้วจึงทำการสเกตช์ภาพขึ้นมาตามคำอธิบายที่ถูกอนุมานไว้ โดยมีรายละเอียดข้อมูลภาพอยู่ทั้งหมด 9 ส่วน คือ เพศ สีผิว โครงหน้า ตา จมูก ปาก หู ทรงผม คิ้ว และหนวดเครา ซึ่งตัวเลือกขององค์ประกอบแต่ละส่วนจะเป็นไปตามตารางที่ 3.1 ยกตัวอย่างของภาพสเกตช์ที่กำหนดรูปแบบองค์ประกอบในส่วนของดวงตาที่มีรูปแบบที่แตกต่างกันได้ดังภาพประกอบที่ 3.4

**ตารางที่ 3.1** องค์ประกอบและหมายเลขกำกับของแต่ละองค์ประกอบ

| | Gender | Skin | Shape | Eyes | Nose | Mouth | Ears | Hair | Eyebrow | Beard |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Male | Light | Square /Triangle | Monolid /hooded | Small | Small | Hidden | None | None /Hidden | None |
| 2 | Female | Fair | Rectangle /Oblong | Almond | Large | Wide /full | Wide /full | Straight-short | Straight | mustache short |
| 3 | | Medium | Round | Round | Wide | Thin | Narrow | Straight-long | Flat | mustache long |
| 4 | | Brown /Red | Oval/Hearth | Downturned | Narrow | Heart shaped | Pointed | Wave-short | Rounded | beard short |
| 5 | | Dark brown | Diamond /Invert Triangle | Upturned | | Heavy upper | Square | Wave-long | Arched | beard long |
| 6 | | Black | | Hooded | | Heavy lower | Round | Curl-short | S-shape | |
| 7 | | | | | | | | Curl-long | | |
| 8 | | | | | | | | Coiled-short | | |
| 9 | | | | | | | | Coiled-long | | |



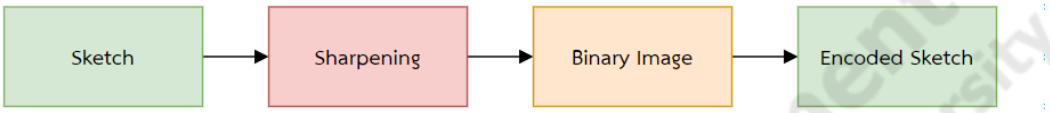

**ภาพประกอบที่ 3.4** ตัวอย่างภาพสเกตช์ภาพจากการกำหนดดวงตาแบบต่าง ๆ

ตัวอย่างภาพสเกตช์ที่ได้กำหนดคำอธิบายภาพเป็น [ Female, Fair skin, Rectangle shape, Almond eyes, Wide nose, Heavy Lower Mouth, Narrow Ears, Wave-long Hair, S-shape Eyebrows, No beard ] หากเปลี่ยนจากข้อมูลตัวอักษรเป็นข้อมูลลำดับของแต่ละองค์ประกอบจะได้เป็นชุดข้อมูล [2, 2, 2, 1, 3, 6, 3, 5, 6, 1] โดยผลลัพธ์ของภาพสเกตช์เป็นไปตามภาพประกอบที่ 3.5

**ภาพประกอบที่ 3.5** ตัวอย่างภาพสเกตช์จากการกำหนดองค์ประกอบ

3.1.2 จัดเตรียมข้อมูลภาพสเกตช์

ภาพสเกตช์ที่ใช้ในการเรียนรู้และทดสอบข้อมูลจะถูกรับเข้ามาเป็นภาพขาว-ดำ (Grayscale) ซึ่งขนาดภาพอาจจะมีขนาดรูปที่หลากหลาย จึงต้องทำการปรับภาพให้เป็นขนาด 1:1 และทำการย่อขนาดลงมาที่ 128*128 pixels เมื่อได้ขนาดภาพที่ต้องการแล้ว จะทำการประมวลผลภาพนโดยมีขั้นตอนการทำงานตามภาพประกอบที่ 3.6 ซึ่งมีรายละเอียดขั้นตอนดังนี้




**ภาพประกอบที่ 3.6** ขั้นตอนการจัดการข้อมูลภาพสเกตช์

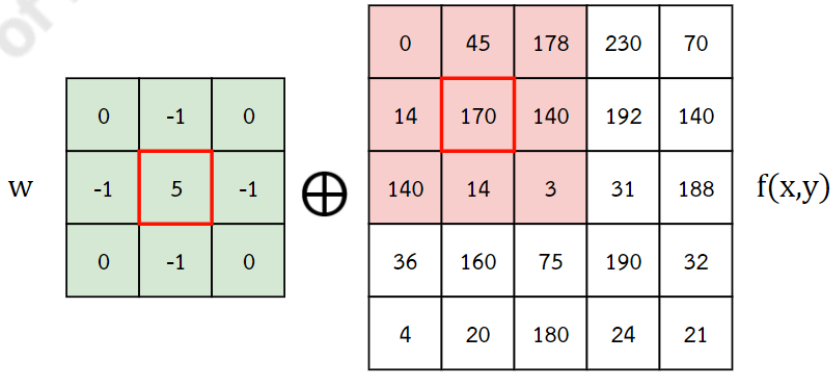
3.1.3 การเพิ่มความคมชัดของภาพ (Image Sharpening)

การเพิ่มความคมชัดของภาพ คือการนำภาพมาทำการคำนวณกับ Filter Matrix โดยใช้การคำนวณแบบ Convolution ซึ่ง Filter Matrix ที่ใช้ จะเป็นไปตาม Matrix (3.2) เพื่อให้ภาพขาว-ดำ (Grayscale) มีความคมชัด ทำให้สามารถแยกลายเส้นได้ง่ายขึ้น ซึ่งสะดวกต่อการประมวลผลในส่วนถัดไป

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.2}$$

$$g(x,y) = w \oplus f(x,y) = \sum_{dx=-a}^{a} \sum_{ay=-b}^{b} w(dx,dy) f(x+dx, y+dy) \tag{3.3}$$

การประมวลผลภาพโดยใช้ Filter Matrix เริ่มจากการกำหนดค่าของ Filter สำหรับการเพิ่มความชัดตาม Matrix (3.2) แล้วจึงนำภาพขาว-ดำ มาทำการคำนวณโดยใช้สมการ (3.3) จากนั้นจะนำข้อมูลของภาพดังภาพประกอบที่ 3.7 (b) ช่องสีแดง มาคำนวณกับ Filter ดังภาพประกอบที่ 3.7 (a)




(a) (b)

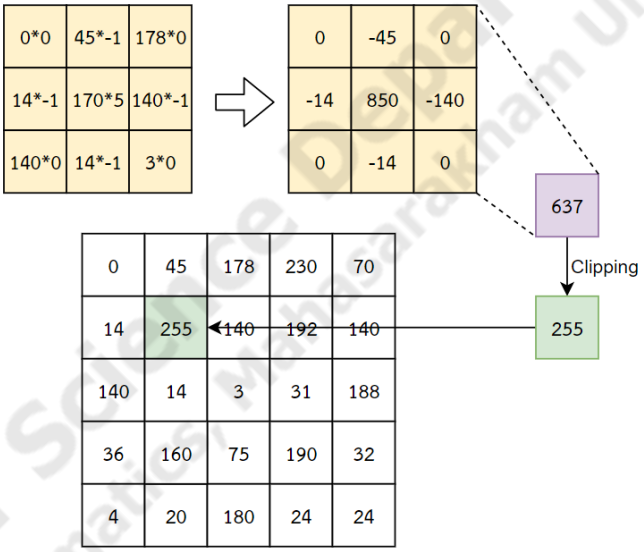
**ภาพประกอบที่ 3.7** ตัวอย่างของตำแหน่งในการคำนวณ Convolution

การคำนวณผลลัพธ์จากภาพประกอบที่ 3.7 สามารถแทนค่าสมการ (3.3) ได้ผลลัพธ์เป็นดังนี้

Output = (0*0)+(45*(-1))+(178*0)+(14*(-1))+(170*5)+(140*(-1))+(140*0)+(14*(-1))+(3*0)

= 637

จะเห็นได้ว่าผลลัพธ์ที่ได้จากการคำนวณนั้นมีค่าสูงกว่าค่าสูงสุดของค่าน้ำหนักสี (ค่าน้ำหนักสีอยู่ระหว่าง [0-255]) จึงต้องทำการปรับค่าของน้ำหนักให้อยู่ในช่วง [0-255] (Clipping) ดังตัวอย่างภาพประกอบที่ 3.8 ซึ่งมีการทำงานคือ หากมีค่าสูงกว่า 255 จะถูกปรับเหลือ 255 และหากมีค่าต่ำกว่า 0 จะถูกปรับให้เป็น 0 เมื่อได้ผลลัพธ์จากการปรับแล้วจึงจะบันทึกค่าลงไป หลังจากนั้นจะประมวลผลต่อไปทีละพิกเซลไปจนครบทุกพิกเซลของภาพ ตัวอย่างของภาพที่ทำ Image sharpening ดังภาพประกอบที่ 3.9




**ภาพประกอบที่ 3.8** การทำ Clipping ก่อนการบันทึกค่า

(a) (b)

(c) (d)

**ภาพประกอบที่ 3.9** ตัวอย่างผลลัพธ์จากการทำ Image Sharpening

3.1.4 การแปลงภาพให้เป็นภาพไบนารี (Image Binarization)

เมื่อได้ภาพที่ถูกเพิ่มความคมชัดของภาพแล้ว ขั้นตอนต่อไปคือทำการแปลงภาพให้เป็นภาพไบนารี โดยใช้วิธี Simple Thresholding ซึ่งจะทำให้ภาพขาว-ดำ กลายเป็นภาพที่มีข้อมูลอยู่เพียง 2 ค่า คือ [0,1]

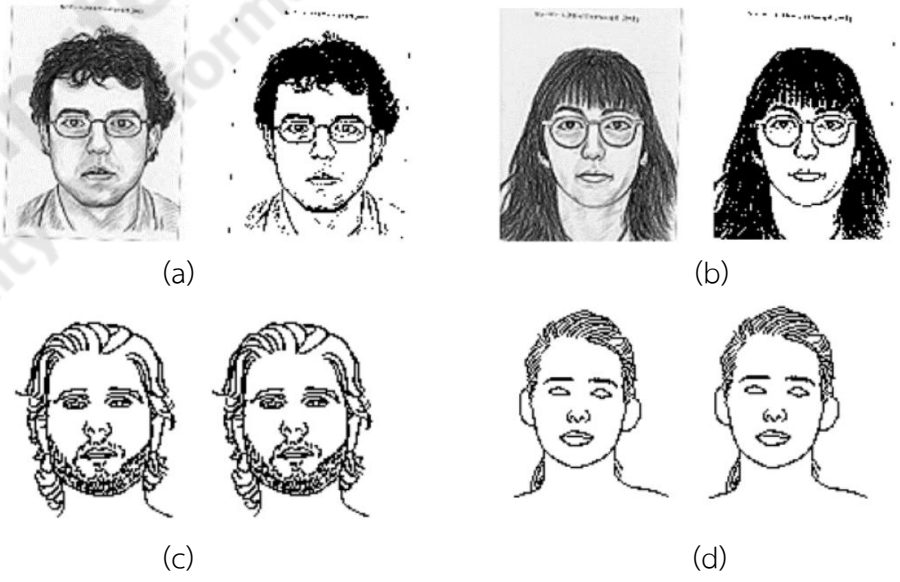
| | | | | |
|---|---|---|---|---|
| 0 | 13 | 250 | 230 | 70 |
| 32 | 245 | 45 | 32 | 29 |
| 250 | 32 | 7 | 0 | 0 |
| 36 | 252 | 11 | 8 | 0 |
| 0 | 20 | 240 | 24 | 0 |

*threshold = 170*

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |

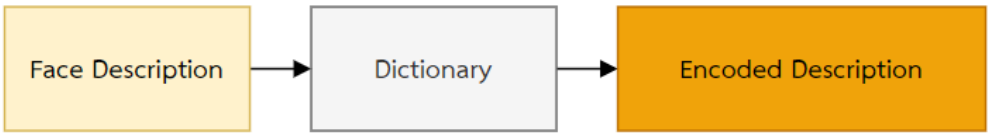
**ภาพประกอบที่ 3.10** การประมวลผล Thresholding

วิธีการทำ Thresholding คือการกำหนดตัวเลข threshold ขึ้นมา โดยที่นี้กำหนดให้เป็น 170 หลังจากนั้นจะนำค่าของแต่ละพิกเซลมาเปรียบเทียบ หากค่าที่ถูกนำมาเปรียบเทียบมีค่ามากกว่าหรือเท่ากับ threshold จะถูกปรับค่าขึ้นไปเป็น 1 (สีขาว) และหากน้อยกว่า threshold จะถูกปรับค่าให้กลายเป็น 0 (สีดำ) ดังภาพประกอบที่ 3.10 (ขวา) ที่แสดงให้เห็นถึงตัวอย่างผลลัพธ์ของข้อมูลที่ผ่านกระบวนการ Thresholding และตัวอย่างของภาพผลลัพธ์เป็นไปตามภาพประกอบที่ 3.11 และหลักจากที่ทำการประมวลผลภาพแล้ว ภาพจะถูกนำไปสร้าง Tensor ขนาด [128, 128, 1] และผลลัพธ์จากการจัดเตรียมข้อมูลภาพสเกตช์นี้จะถูกเรียกว่า สเกตช์ที่ถูกเข้ารหัส (Encoded sketch)




**ภาพประกอบที่ 3.11** ผลลัพธ์จากการทำ Image Binarization

3.1.5 จัดเตรียมข้อมูลคำอธิบายภาพสเกตช์




**ภาพประกอบที่ 3.12** ขั้นตอนการจัดการคำอธิบายภาพสเกตช์

ในส่วนของข้อมูลคำอธิบายภาพของภาพสเกตช์ จะอยู่ในรูปแบบของ Vector ที่เก็บข้อมูลองค์ประกอบของภาพ มีขนาดความยาว 10 ช่อง โดยแต่ละช่องจะเก็บข้อมูลเป็นตำแหน่ง (Index) ขององค์ประกอบในแต่ละหมวด ตัวเลือกของแต่ละหมวดจะเป็นไปตามตารางที่ 3.2 ซึ่งมีรูปแบบตัวเลือกขององค์ประกอบทั้งหมด 55 ตัวเลือก

**ตารางที่ 3.2** ตัวเลือกขององค์ประกอบในแต่ละหมวด

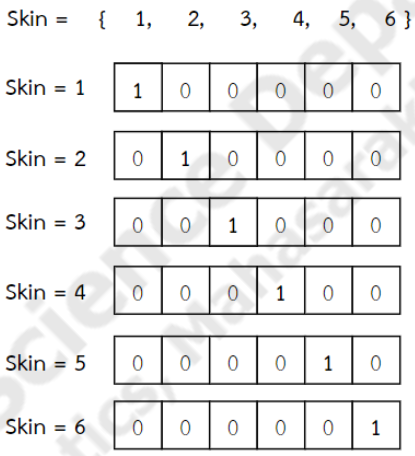
| Category | Items | Length |
|---|---|---|
| Gender | Male, Female | 2 |
| Skin | Light, Fair, Medium, Brown/Red, Dark Brown, Black | 6 |
| Shape | Square/Triangle, Rectangle/Oblong, Round, Oval/ Hearth, Diamond | 5 |
| Eyes | Monolid, Almond, Round, Downturned, Upturned, Hooded | 6 |
| Nose | Small, Large, Wide, Narrow | 4 |
| Mouth | Small, Wide/Full, Thin, Heart shaped, Heavy upper, Heavy lower | 6 |
| Ears | Hidden, Wide/Full, Narrow, Pointed, Square, Round | 6 |
| Hair | None, Straight-short, Straight-long, Wave-short, Wave-long, Curl-short, Curl-long, Coiled-short, Coiled-long | 9 |
| Eyebrows | None/Hidden, Straight, Flat, Rounded, Arched, S-shape | 6 |
| Bread | None, mustache short, mustache long, beard short, beard long | 5 |
| | **Total** | 55 |

**ตารางที่ 3.3** ตัวอย่างของข้อมูลคำอธิบายภาพสเกตช์

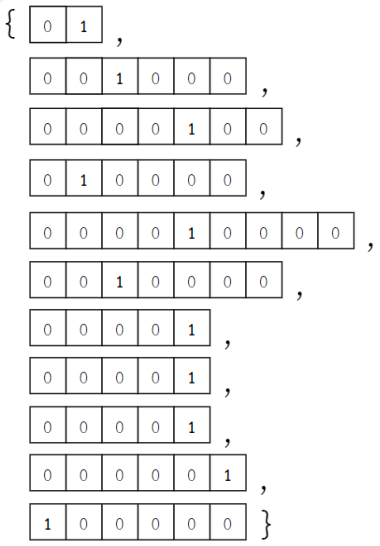
| filename | gender | skin | ears | eyes | nose | mouth | shape | hair | eyebrows | bread |
|---|---|---|---|---|---|---|---|---|---|---|
| C_001_01 | 2 | 3 | 5 | 2 | 3 | 5 | 5 | 5 | 6 | 1 |
| C_002_01 | 2 | 1 | 1 | 3 | 3 | 4 | 1 | 2 | 2 | 1 |
| C_003_01 | 2 | 2 | 1 | 1 | 6 | 3 | 1 | 3 | 5 | 1 |

ในการจัดการคำอธิบายให้เหมาะสมกับการเรียนรู้ของโมเดล จะใช้วิธีการสร้างเวกเตอร์คำอธิบายที่ถูกเข้ารหัส (Encoded description vector) จากข้อมูลคำอธิบายภาพสเกตช์ โดยมีขั้นตอนการประมวลผลดังนี้

1. เข้ารหัสสมาชิกทุกตัวด้วยวิธีการ One-Hot Encoding โดยนำค่าองค์ประกอบของแต่ละหมวดมาสร้าง Vector เพื่อแสดงค่าขององค์ประกอบในรูปแบบค่าบิต (Bit Representation) โดยใช้วิธีการเข้ารหัสแบบ One-Hot Encoding เพื่อให้ได้ผลลัพธ์ที่เป็นรูปแบบไบนารี (Binary Values) ภาพประกอบที่ 3.13 แสดงถึงตัวอย่างของผลลัพธ์จากการเข้ารหัสขององค์ประกอบหมวด Skin เมื่อมีค่า 1 ถึง 6 ยกตัวอย่างการประมวลผลโดยเริ่มจากการกำหนดข้อมูลขององค์ประกอบเป็น [2, 3, 5, 2, 4, 3, 5, 5, 5, 5, 1] แล้วจึงนำไปประมวลผล เมื่อโมดูลทำการเข้ารหัสครบทุกสมาชิก จะได้ผลลัพธ์เป็นชุดข้อมูลดังตัวอย่างภาพประกอบที่ 3.14 แสดงให้เห็นถึงผลลัพธ์การเข้ารหัสของแต่ละองค์ประกอบให้อยู่ในรูปของค่าบิต




**ภาพประกอบที่ 3.13** ตัวอย่างผลลัพธ์การเข้ารหัสด้วยวิธี One-Hot Encoding



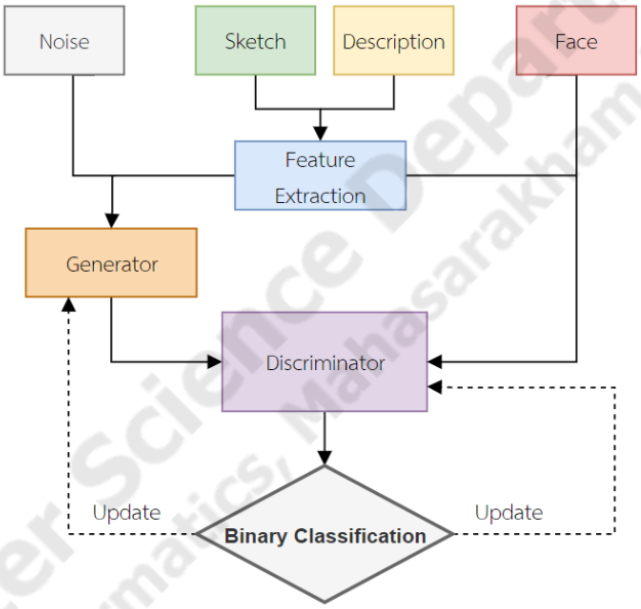

**ภาพประกอบที่ 3.14** ตัวอย่างผลลัพธ์ที่ถูกเข้ารหัสครบทุกสมาชิก

2. เชื่อมต่อข้อมูลที่ถูกเข้ารหัส โดยนำผลลัพธ์ของข้อมูลองค์ประกอบที่ถูกเข้ารหัสทุกตัวจากภาพประกอบที่ 3.14 มาเชื่อมต่อเข้าด้วยกัน (Concatenate) เพื่อสร้างเป็นเวกเตอร์คำอธิบายองค์ประกอบที่ถูกเข้ารหัส (Encoded description vector) ที่มีขนาดความยาว 55 ตัว ซึ่งจะได้ผลลัพธ์ออกมาดังนี้

Encoded description vector = [ 0,1,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,1,0,0,
0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0 ]

3. ข้อมูลทั้งในส่วนของ Encoded sketch และ Encoded description vector จะถูกนำไปใช้เป็น Input สำหรับโมเดลที่จะกล่าวถึงในขั้นต่อไป

## 3.2 ขั้นตอนการพัฒนาโมเดล




**ภาพประกอบที่ 3.15** ขั้นตอนการทำงานของโมเดล CGAN

ในการพัฒนาโมเดลการสร้างภาพใบหน้าจำลอง จะใช้รูปแบบการเรียนรู้แบบ Conditional Generative Adversarial Network หรือ CGAN ร่วมกับ Hyper-Parameter พิเศษของระบบงานนี้ ซึ่งก็คือ ”คำอธิบายภาพสเกตช์” โดยขั้นตอนการเรียนรู้ของโมเดลเป็นไปตามภาพประกอบที่ 3.15 ซึ่งข้อมูลที่ใช้สำหรับการประมวลผลของระบบนี้มีทั้งหมด 4 ส่วน คือ ภาพจุดรบกวน (Noise) คำอธิบายของภาพสเกตช์ที่ถูกเข้ารหัสแล้ว (Encoded description vector), ภาพใบหน้าจริง (Real face) และภาพสเกตช์ของภาพใบหน้าจริงที่ถูกเข้ารหัส (Encoded sketch) ซึ่งในการเรียนรู้ของโมเดลมีขั้นตอนการทำงานดังนี้

1. Feature Extraction Module คือโมดูลที่ทำหน้าที่สกัดคุณลักษณะเฉพาะออกมาจากคู่ภาพสเกตช์และคำอธิบายภาพสเกตช์ ซึ่งจะได้ผลลัพธ์ออกมาเป็นเวกเตอร์ที่ถูกเข้ารหัส (Encoded Vector) ขนาดความยาว 128 โดยมี Input 2 ส่วน คือ Encoded description และ Encoded sketch

2. Generator Model คือโมเดลที่ทำหน้าที่จำลองภาพใบหน้า ซึ่งจะได้ผลลัพธ์เป็นภาพใบหน้าจำลอง รูปแบบ RGB ขนาด 128*128 pixels (Generated face) ซึ่งมี Input อยู่ทั้งหมด 2 ส่วน คือ ภาพจุดรบกวน (Noise) และ Encoded vector

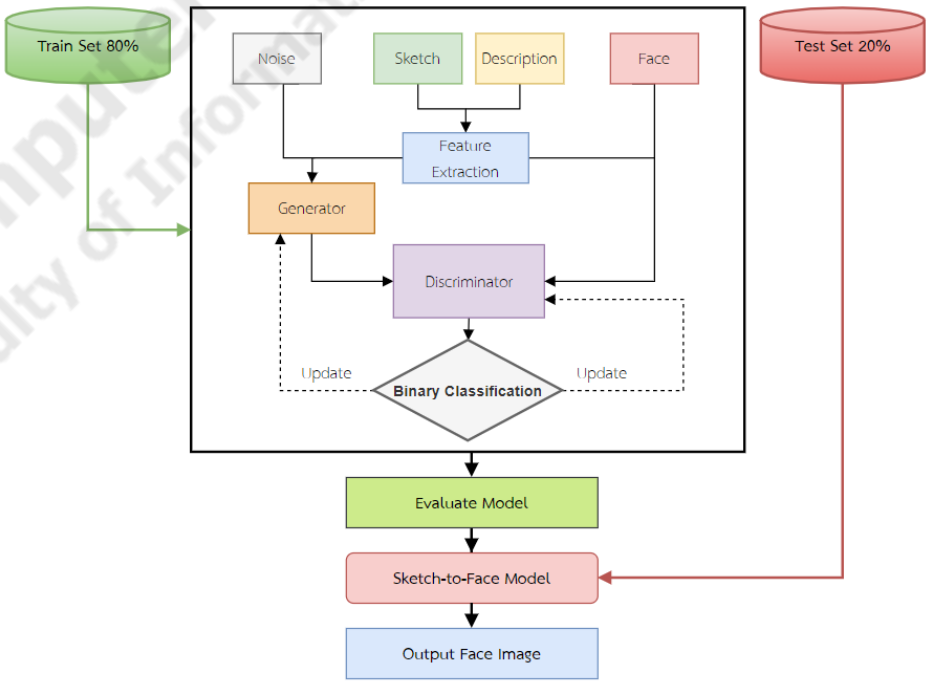
3. Discriminator Model คือโมเดลที่ทำหน้าที่เปรียบเทียบและตัดสินภาพ เพื่อคัดแยกภาพที่ถูกสร้างขึ้นมาจาก Generator Model และภาพใบหน้าจริง มี Input อยู่ทั้งหมด ส่วน คือ Encoded vector, Real face หรือ Generated face ซึ่งผลลัพธ์จาก Discriminator จะเป็นค่าความเป็นไปได้ของภาพว่าเป็นภาพจริง หรือภาพที่ถูกสร้างขึ้น ซึ่งมีค่าอยู่ระหว่างช่วง [0,1]

4. เมื่อได้ผลลัพธ์จาก Generator และ Discriminator ออกมาแล้ว จะทำการคำนวณค่า Loss ของโมเดลของทั้งสองโมเดลโดยใช้วิธีการคำนวณ Binary Cross-Entropy หรือก็คือการตรวจสอบค่า Loss โดยจะมีบทลงโทษหากการทำนายผิดพลาด ซึ่งจะส่งผลให้เกิดค่า Loss ที่สูงขึ้น สิ่งที่ใช้ในการคำนวณของโมเดลแต่ละตัว มีดังนี้

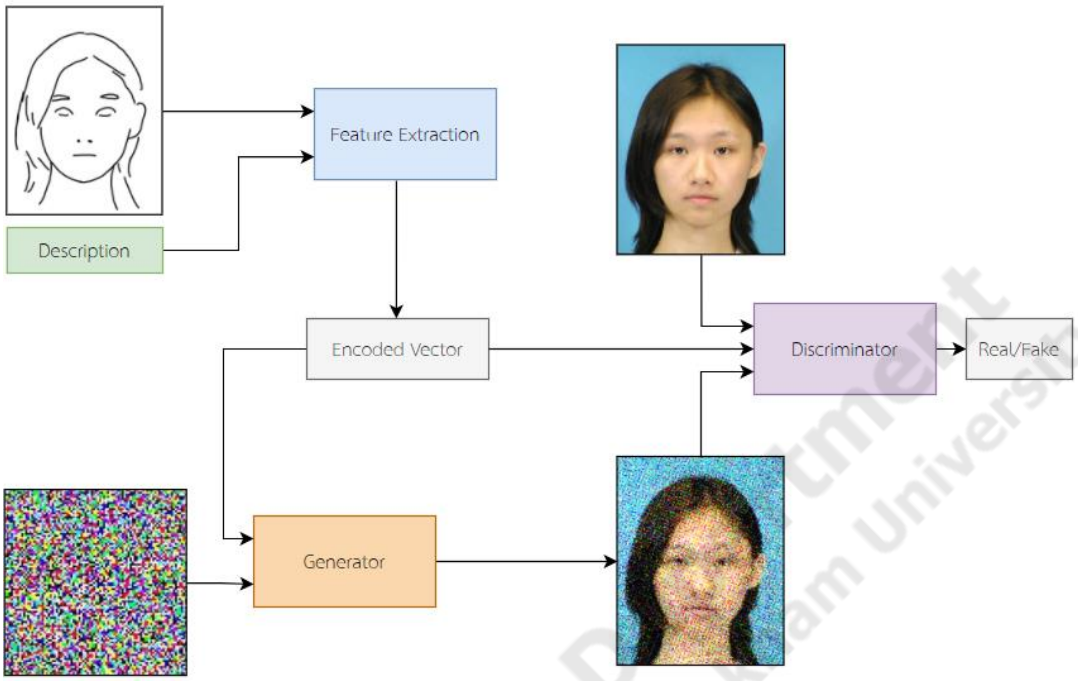
- Generator คำนวณโดยใช้เวกเตอร์เข้ารหัสและภาพใบหน้าที่ถูกจำลองออกมา
- Discriminator คำนวณค่า Loss สองครั้ง คือการทำนายผลของภาพใบหน้าจริงเทียบกับผลเฉลย และการทำนายผลของภาพใบหน้าจำลองเทียบกับผลเฉลย

5. หลังจากที่ได้ค่า Loss มาแล้ว โมเดลจะทำการปรับปรุง Hyper-Parameter จากค่า Loss เพื่อให้การประมวลผลในรอบถัดไปนั้นได้ผลลัพธ์ที่ดียิ่งขึ้น โดยแต่ละโมเดลจะมีเทคนิคการปรับปรุงที่แตกต่างกัน คือ Generator จะทำการลดระดับ (Minimize) ค่า Loss ให้ได้มากที่สุด เพื่อให้ภาพที่จะถูกสร้างขึ้นในรอบถัดไปนั้นมีความใกล้เคียงกับภาพเฉลยให้ได้มากที่สุด ส่วนของ Discriminator จะทำการเพิ่มระดับ (Maximize) ค่า Loss เพื่อปรับปรุงการตรวจสอบภาพเท็จให้มีประสิทธิภาพมากขึ้น

### 3.3 ขั้นตอนการเรียนรู้และทดสอบโมเดล



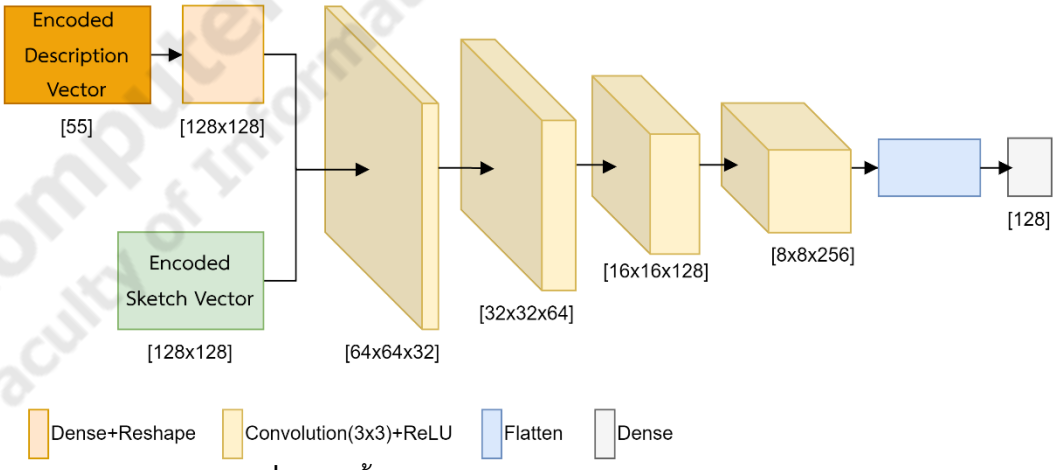

**ภาพประกอบที่ 3.16** กรอบการดำเนินงาน




**ภาพประกอบที่ 3.17** แสดงการทำงานของโมเดล CGAN

ในส่วนของการเรียนรู้และทดสอบโมเดล มีลำดับการทำงานตามภาพประกอบที่ 3.16 โดยจะใช้สัดส่วนการแบ่งข้อมูลอยู่ที่ 80/20 หรือ ส่วนของชุดการเรียนรู้ (Train Set) 80% และส่วนของชุดการทดสอบ (Test Set) 20% ที่จะถูกนำไปใช้ในการประเมินประสิทธิภาพโมเดล

3.3.1 การทำงานของโมดูลการแยกองค์ประกอบ




**ภาพประกอบที่ 3.18** ขั้นตอนการทำงานของ Feature extraction module

ในส่วนของโมดูลการคัดแยกคำอธิบาย (Feature extraction module) จะทำการค้นหาความสัมพันธ์ของของภาพสเกตช์ (Tensor ขนาด [128,128,1]) และคำอธิบายภาพสเกตช์ (Vector ขนาด 62) โดยเริ่มจากการนำคำอธิบายภาพสเกตช์ที่ถูกเข้ารหัสมาสร้างให้เป็น Tensor ขนาด [128,128,1] โดยมีขั้นตอนดังนี้

1. นำคำอธิบายที่ถูกเข้ารหัสมาสร้างชั้นฝังข้อมูล (Embedding layer) เพื่อให้ได้ข้อมูลที่อยู่ในรูปแบบ Matrix ขนาด (62, 128)

2. สร้างชั้นปรับความหนาแน่น (Dense layer) ที่มีจำนวนโหนดเท่ากับ 16,384 โหนด (128x128) จากผลลัพธ์ที่ได้จากชั้นฝังข้อมูล จากนั้นทำการเปลี่ยนแปลงรูปร่าง (Reshape) Vector ขนาด 16,384 ให้อยู่ในรูป Tensor ขนาด [128, 128, 1]

3. นำผลลัพธ์จากการสร้าง Tensor โดยใช้คำอธิบายที่ถูกเข้ารหัสมาเชื่อมต่อเข้าด้วยกัน (Concatenate) กับภาพสเกตช์ที่ถูกเข้ารหัส จะได้ผลลัพธ์ออกมาเป็น Tensor ขนาด [128, 128, 2]

4. นำ Tensor ขนาด [128, 128, 2] มาเข้ากระบวนการ Convolution เพื่อค้นหาคุณลักษณะเด่นของคู่ภาพสเกตช์และคำอธิบายภาพ โดยจะประมวลผลทั้งสิ้น 4 ชั้น คือชั้น 32, 64, 128, 256 ตามลำดับ ได้ผลลัพธ์ออกมาเป็น Tensor ขนาด [8, 8, 256]

5. ทำการ Flatten ข้อมูล Tensor เพื่อให้ได้ข้อมูล Vector ขนาด 16,384 (8x8x256)

6. นำข้อมูล Vector เข้าชั้นปรับความหาแน่น เพื่อให้ได้ผลลัพธ์ Vector ที่มีขนาด 128

3.3.2 ตัวอย่างการประมวลผล Convolution

$$n_{out} = \left[\frac{n_{in} + 2p - k}{2}\right] + 1 \qquad (3.4)$$

การทำ Convolution เริ่มด้วยการกำหนดขนาดมิติฟิลเตอร์ (Filter dimension) เท่ากับ 128, ขนาดแกน (Kernel, k) เท่ากับ 4, การเคลื่อนของแกน (Stride, s) เท่ากับ 2, กำหนดค่าการขยาย (Padding, p) เท่ากับ “Same” (ในที่นี้กำหนดให้เป็น 1) และภาพที่ถูกป้อนเข้ามามีขนาด [64, 64, 1] โดยใช้สมการ (3.4) คำนวณในแต่ละแกน (ด้านกว้างและด้านสูง) ได้ผลลัพธ์เป็น Tensor ขนาด [32, 32, 128] ซึ่งสามารถแทนค่าสมการได้ดังนี้

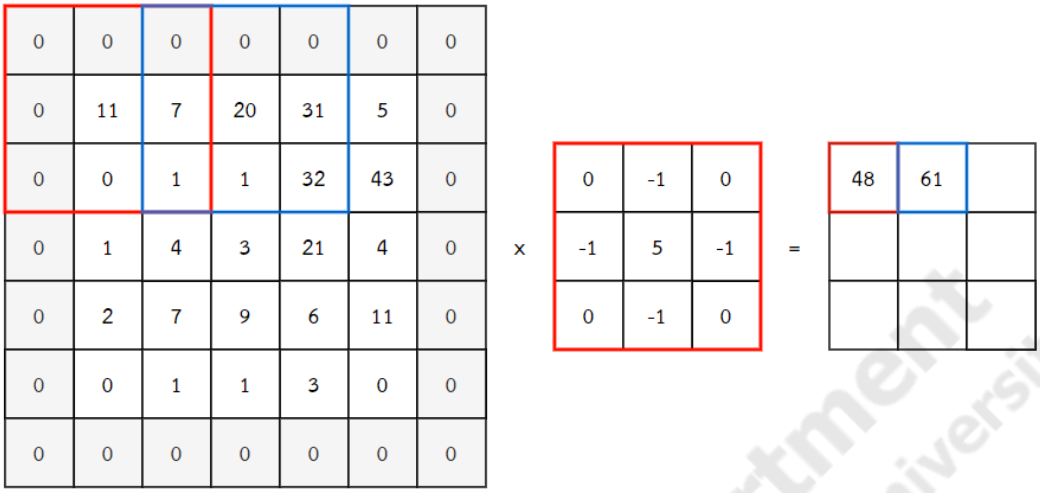
$$width_{out} = \left[\frac{64 + 2(1) - 4}{2}\right] + 1 \ = 32$$

$$height_{out} = \left[\frac{64 + 2(1) - 4}{2}\right] + 1 = 32$$

ตัวอย่างของการคำนวณค่าผลลัพธ์ของ Feature map ที่ได้จากการทำ Convolution โดยกำหนดค่า kernel = 3, stride = 2 และ padding = 1 ซึ่งจะเริ่มด้วยการวาง Kernel ลงไปบนภาพเริ่มต้นจากมุมซ้าย-บน จากนั้นทำการขยับไปทีละ 2 ช่อง ในแนวตั้ง จนครบทุกช่อง ภาพประกอบที่ 3.19 แสดงถึงตัวอย่างการคำนวณของ Feature map ช่องที่ (0,0) และ (0,1) โดยใช้สมการ (3.3) ได้ผลลัพธ์ดังนี้

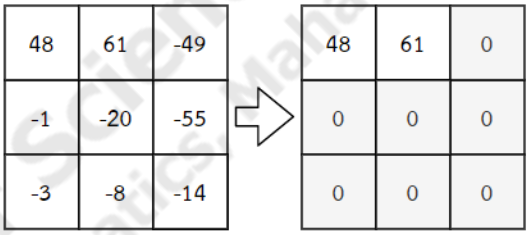
$F_{(0,0)}$ = (0*0)+(0*(-1))+(0*0)+(0*(-1))+(11*5)+(7*(-1))+(0*0)+(0*(-1))+(1*0)
= 48

$F_{(0,1)}$ = (0*0)+(0*(-1))+(0*0)+(7*(-1))+(20*5)+(31*(-1))+(1*0)+(1*(-1))+(32*0)
= 61




**ภาพประกอบที่ 3.19** ตัวอย่างผลลัพธ์จากการทำ Convolution

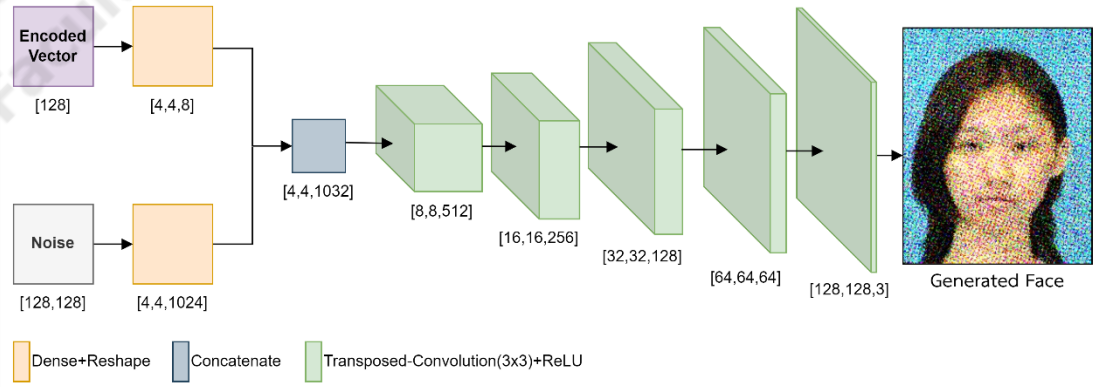
หลักจากที่คำนวณ Feature maps ออกมาเสร็จสมบูรณ์แล้ว (ภาพประกอบที่ 3.20 ซ้าย) จะถูกนำไปผ่านฟังก์ชัน ReLU (Rectified Linea Unit) เพื่อทำการ Normalize ให้ค่าของแต่ละพิกเซลอยู่ในช่วง [0,255] โดยการปรับค่าที่เป็นค่าลบ (Negative values) ให้เท่ากับ 0 (ภาพประกอบที่ 3.20 ขวา)




**ภาพประกอบที่ 3.20** ผลลัพธ์จากการทำ ReLU

จากนั้นโมดูลนี้จะทำการประมวลผลและปรับค่าน้ำหนักเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดออกมาสำหรับใช้เป็นข้อมูลในการประมวลผลของโมเดล Generator และ Discriminator ที่จะกล่างถึงต่อไป

3.3.3 การทำงานของโมเดลการจำลองภาพ




**ภาพประกอบที่ 3.21** ขั้นตอนการทำงานของ Generator Model

โมเดลการจำลองภาพ (Generator Model) คือส่วนในการสร้างภาพใบหน้าจำลองจาก Input โดยจะใช้เทคนิค Transposed-Convolution ของข้อมูล Input 2 ส่วน ส่วนแรกคือภาพจุดรบกวน (Noise) เป็นภาพที่ถูกสร้างจากการสุ่มค่าตัวเลข เป็น Tensor ขนาด [128, 128, 1] และส่วนที่สองคือผลลัพธ์จากโมดูลการคัดแยกคำอธิบาย ที่เป็นเวกเตอร์คำอธิบายภาพสเกตช์ที่ถูกเข้ารหัส (Encoded description vector) ซึ่งมีขนาดความยาว 128 โดยขั้นตอนการประมวลผลของโมเดลการจำลองภาพมีขั้นตอนการทำงานดังนี้

1. นำ Noise Tensor ขนาด [128,128,1] ไปเข้าชั้นปรับความหนาแน่น ที่มีจำนวนโหนดเท่ากับ 16,384 โหนด (4x4x1024) จากนั้นจะถูกเปลี่ยนแปลงขนาด (Reshape) ให้เป็น Tensor ขนาด [4, 4, 1024]
2. ในส่วนของเวกเตอร์เข้ารหัส จะถูกนำไปเข้าชั้นปรับความหนาแน่นที่มีจำนวนโหนดเท่ากับ 128 โหนด (4x4x8) จากนั้นจะทำการปรับขนาดให้อยู่ในรูป Tensor [4, 4, 8]
3. นำข้อมูลที่ได้จาก 1) และ 2) มาเชื่อมเข้าด้วยกัน เพื่อสร้าง Tensor [4, 4, 1032]
4. นำ Tensor เข้ากระบวนการ Transposed-Convolution โดยจะประมวลผลทั้งสิ้น 5 ชั้น คือชั้น 512, 256, 128, 64 และ 3 ตามลำดับ ได้ผลลัพธ์ออกมาเป็น Tensor [128, 128, 3]
5. คำนวณ Loss ของ Generator โดยใช้เทคนิค Binary cross-entropy

3.3.4 ตัวอย่างการประมวลผล Transposed-Convolution

$$n_{out} = (n_{in} - 1) \cdot s - 2 \cdot p + (k - 1) + 1 \qquad (3.5)$$

การทำ Transposed-Convolution Convolution เริ่มด้วยการกำหนดขนาดมิติฟิลเตอร์ (Filter dimension) เท่ากับ 128, ขนาดแกน (Kernel, k) เท่ากับ 4, การเคลื่อนของแกน (Stride, s) เท่ากับ 2, กำหนดค่าการขยาย (Padding, p) เท่ากับ “Same” (ในที่นี้กำหนดให้เป็น 1) และภาพที่ถูกป้อนเข้ามามีขนาด [32, 32, 1] โดยใช้สมการ (3.5) คำนวณในแต่ละแกน (ด้านกว้างและด้านสูง) ได้ผลลัพธ์เป็น Tensor ขนาด [64, 64, 128] ซึ่งสามารถแทนค่าสมการได้ดังนี้

$$width_{out} = (32 - 1) \cdot 2 - 2 \cdot 1 + (4 - 1) + 1 \ = 64$$

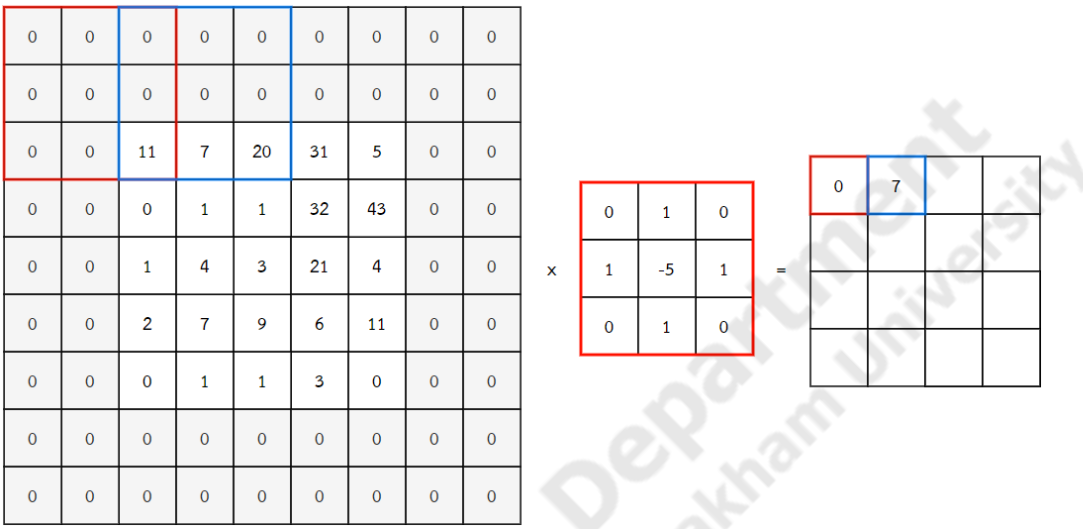
$$height_{out} = (32 - 1) \cdot 2 - 2 \cdot 1 + (4 - 1) + 1 \ = 64$$

ตัวอย่างของการคำนวณค่าผลลัพธ์ของ Feature map ที่ได้จากการทำ Transposed-Convolution (k = 3, s = 2, p = 2) เริ่มด้วยการวางทาง kernel ลงไปบนภาพ เริ่มต้นจากมุมซ้าย-บนแล้วจึงขยับไปทีละ 2 ช่อง ทั้งในแนวตั้งและแนวนอน ยกตัวอย่างการคำนวณ Feature map ดังภาพประกอบที่ 3.22 ช่องที่ (0,0) และ (0,1) โดยใช้สมการ (3.3) ได้ผลลัพธ์ดังนี้

$F_{(0,0)}$ = (0*0)+(0*1)+(0*0)+(0*1)+(0*(-5))+(0*1)+(0*0)+(0*1)+(11*0)
= 0

$F_{(0,1)}$ = (0*0)+(0*1)+(0*0)+(0*1)+(0*(-5))+(0*1)+(11*0)+(7*1)+(20*0)
= 7

หลักจากที่คำนวณ Feature maps ออกมาเสร็จสมบูรณ์แล้ว จะถูกนำไปผ่านฟังก์ชัน ReLU เพื่อทำการ Normalize ให้ค่าของแต่ละพิกเซลอยู่ในช่วง [0,255] โดยการปรับค่าที่เป็นค่าลบ (Negative values) ให้เท่ากับ 0




**ภาพประกอบที่ 3.22** ตัวอย่างผลลัพธ์จากการทำ Transposed-Convolution

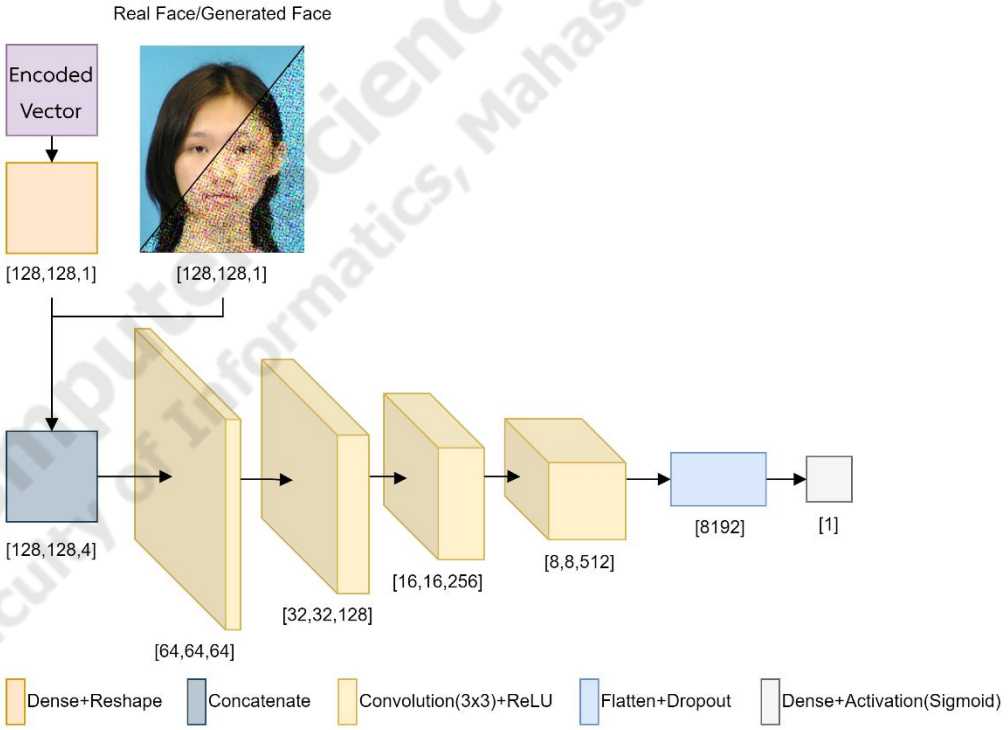
(a) (b)

(c) (d)

**ภาพประกอบที่ 3.23** ตัวอย่างผลลัพธ์ของ Generator Epoch 10 60 120 180 ตามลำดับ


**ภาพประกอบที่ 3.24** ตัวอย่างผลลัพธ์ (b) และภาพเฉลย (c)
จากการส่งภาพสเกตช์เข้าไป (a)

3.3.5 การทำงานของโมเดลการตรวจสอบภาพ




**ภาพประกอบที่ 3.25** ขั้นตอนการทำงานของ Discriminator Model

โมเดลการตรวจสอบภาพ (Discriminator Model) เป็นโมเดลที่ทำหน้าที่ในการตรวจสอบภาพใบหน้าที่ถูกสร้างขึ้นมาจาก Generator โดยจะมีข้อมูล Input อยู่ทั้งหมด 2 ส่วนคือ เวกเตอร์เข้ารหัสภาพใบหน้าจริงหรือภาพใบหน้าจำลอง โดยส่วนการทำงานของ Discriminator มีขั้นตอนดังนี้

1. ภาพใบหน้าจำลองที่เป็นภาพสี (RGB Image) จะถูกเปลี่ยนให้กลายเป็น Tensor ขนาด [128, 128, 3]

2. เวกเตอร์เข้ารหัส จะถูกนำไปเข้าชั้นปรับความหนาแน่นที่มีจำนวนโหนดเท่ากับ 16,384 โหนด (128x128) จากนั้นจะทำการปรับของผลลัพธ์เวกเตอร์จากชั้นปรับความหนาแน่นให้อยู่ในรูป Tensor ขนาด [128, 128, 1]

3. นำข้อมูลที่ได้จาก 1 และ 2 เชื่อมเข้าด้วยกัน เพื่อให้ได้ Tensor ขนาด [128, 128, 4]

4. นำ Tensor เข้ากระบวนการ Convolution โดยจะประมวลผลทั้งหมด 4 ชั้น คือชั้น 64, 128, 256 และ 512 ตามลำดับ ได้ผลลัพธ์ออกมาเป็น Tensor ขนาด [4, 4, 512]

5. นำ Tensor เข้าชั้น Flatten เพื่อให้ได้ข้อมูล Vector ขนาด 8,192 (4x4x512)

6. นำ Vector เข้าชั้นปรับความหนาแน่นให้กลายเป็น Scalar โดยใช้ Activation ที่ชื่อ Sigmoid เพื่อให้ได้ผลลัพธ์ที่มีค่าระหว่าง [0-1] สำหรับใช้ในการทำนายความน่าจะเป็น

3.3.6 การคำนวณค่า Loss

ค่า Loss ถูกคำนวณเพื่อนำค่าที่ได้ไปทำการปรับปรุงโมเดล ซึ่งจะทำการปรับปรุงทั้งของ Generator และ Discriminator เพื่อให้การเรียนรู้ในรอบต่อไปมีคุณภาพที่ดียิ่งขึ้น โดยการคำนวณค่า Loss จะใช้วิธีการคำนวณแบบ Binary Cross-Entropy [12] โดยใช้ฟังก์ชัน (3.6) ซึ่งสามารถเขียนให้อยู่ในรูปแบบของสมการดังสมการ (3.7) และค่า Loss ของทั้งระบบสามารถคำนวณได้โดยใช้สมการ (3.8)

$$J(p,y) = \begin{cases} -y \cdot \log p & y = 1 \\ -(1-y) \cdot \log(1-p) & y = 0 \end{cases} \tag{3.6}$$

$$J(p,y) = -(y \cdot \log p + (1-y) \cdot \log(1-p)) \tag{3.7}$$

$$J(w) = -\frac{1}{N}\sum_{n=1}^{N}[y_n \cdot \log \hat{y}_n + (1-y_n) \cdot \log(1-\hat{y}_n)] \tag{3.8}$$

ในส่วนของฟังก์ชัน (3.6) จะมีการทำงานอยู่ 2 รูปแบบ หากภาพนั้นเป็นภาพจริง ($y = 1$) จะถูกคำนวณด้วยสมการ $-y \cdot \log p$ และหากภาพเป็นภาพที่ถูกสร้างขึ้น ($y = 0$) จะถูกคำนวณด้วยสมการ $-(1-y) \cdot \log(1-p)$

ขั้นตอนการคำนวณ Loss เพื่อปรับปรุงตัวโมเดลนั้นจะเริ่มจากการปรับปรุงตัวของ Discriminator ซึ่งจะคำนวณค่า Loss ทั้งหมด 2 ครั้งแรกจะคำนวณค่า Loss ของภาพใบหน้าจริง และครั้งที่สองจะคำนวณค่า Loss ของภาพใบหน้าจำลอง โดยหลักการของ Discriminator คือการยอมรับภาพจริงทุก ๆ ภาพ และปฏิเสธภาพใบหน้าจำลองทุก ๆ ภาพ แม้ภาพจำลองนั้นจะถูกจำลองออกมาได้คล้ายคลึงกับภาพจริงหรือมีแนวโน้มที่ตรงตามคำอธิบายก็ตาม

3.3.7 ตัวอย่างการคำนวณค่า Loss ของ Discriminator

1. การคำนวณครั้งที่ 1กำหนดให้ภาพใบหน้านั้นเป็นภาพจริง ($y = 1$) และคำนวณความน่าจะเป็นของภาพใบหน้าจริงโดยเทียบกับคำอธิบายของภาพ ได้ผลลัพธ์ออกมาเป็น 0.9 ($\hat{y} = 0.9$) แทนค่าในฟังก์ชัน (3.9) ได้ผลลัพธ์ดังนี้

$$J(0.9, 1) = -1 \cdot \log(0.9)$$

$$J(0.9, 1) = 0.0457$$

2. การคำนวณครั้งที่ 2 กำหนดให้ภาพใบหน้าเป็นภาพจำลอง ($y = 0$) และคำนวณความน่าจะเป็นของภาพใบหน้าจำลองโดยเทียบกับคำอธิบายของภาพ ได้ผลลัพธ์ออกมาเป็น 0.75 ($\hat{y} = 0.75$) แทนค่าในฟังก์ชัน (3.10) ได้ผลลัพธ์ดังนี้

$$J(0.75, 0) = -(1 - 0) \cdot \log(1 - 0.75)$$

$$J(0.75, 0) = -1 \cdot \log(0.25)$$

$$J(0.75, 0) = 0.6020$$

เมื่อทำการปรับปรุงคุณภาพการตรวจสอบภาพของ Discriminator แล้ว จะทำการคำนวณค่า Loss ของ Generator โดยอ้างอิงจากค่าความผิดพลาดของ Discriminator หมายความว่าตัว Generator จะทำการคำนวณค่า Loss โดยใช้ภาพที่ถูกจำลองขึ้น และกำหนดให้ภาพจำลองนั้นเป็นภาพจริง ($y = 1$)

3.3.8 ตัวอย่างการคำนวณค่า Loss ของ Generator

กำหนดให้ค่าจริงของภาพเท่ากับ 1 ($y = 1$) และกำหนดความน่าจะเป็นของภาพจำลองที่มีต่อคำอธิบายภาพเท่ากับ 0.6 ($\hat{y} = 0.6$) สามารถแทนค่าฟังก์ชัน (3.11) ได้ดังนี้

$$J(0.6, 1) = -1 \cdot \log(0.6)$$

$$J(0.6, 1) = 0.2218$$

**ตารางที่ 3.4** ผลลัพธ์จากการคำนวณค่า Loss

| C_001_01 | $y$ | $\hat{y}$ | $J(\hat{y}, y)$ | Log |
|---|---|---|---|---|
| Discriminator (Real) | 1 | 0.9 | $-1 \cdot \log(0.9)$ | 0.0457 |
| Discriminator (Fake) | 0 | 0.75 | $(1 - 0) \cdot \log(1 - 0.75)$ | 0.602 |
| Generator | 1 | 0.6 | $-1 \cdot \log(0.6)$ | 0.2218 |

จากตารางที่ 3.4 ค่า Loss ของทั้งระบบสามารถคำนวณได้โดยใช้สมการ (3.8) สามารถแทนค่าและทำการคำนวณออกมาได้ดังนี้

$$J(w) = -\frac{1}{3} \cdot [(-0.0457) + (-0.602) + (-0.2218)]$$

$$J(w) = -\frac{1}{3} \cdot (-0.8695)$$

$$J(w) = -\frac{(-0.8695)}{3}$$

$$J(w) = 0.2898$$

**3.4 การวัดประสิทธิภาพ**

ในส่วนของการวัดประสิทธิภาพของระบบจะทำการประเมินทั้งหมด 3 ส่วน คือ การประเมินประสิทธิภาพในการเรียนรู้ของตัวโมเดล และการวัดคุณภาพของภาพผลลัพธ์ที่ถูกสร้างขึ้นมาจากตัว Generator Model และการประเมินประสิทธิภาพของแอปพลิเคชัน

3.4.1 การประเมินประสิทธิภาพในการเรียนรู้ของโมเดล

ในการประเมินประสิทธิภาพในการเรียนรู้ จะทำการประเมิน Validation Loss โดยการนำผลลัพธ์ที่ได้จากการคำนวณค่า Loss โดยใช้เทคนิค Binary Cross-entropy มาวิเคราะห์ประสิทธิภาพและแนวโน้มในการเรียนรู้ของตัวโมเดล

3.4.2 การประเมินคุณภาพของผลลัพธ์

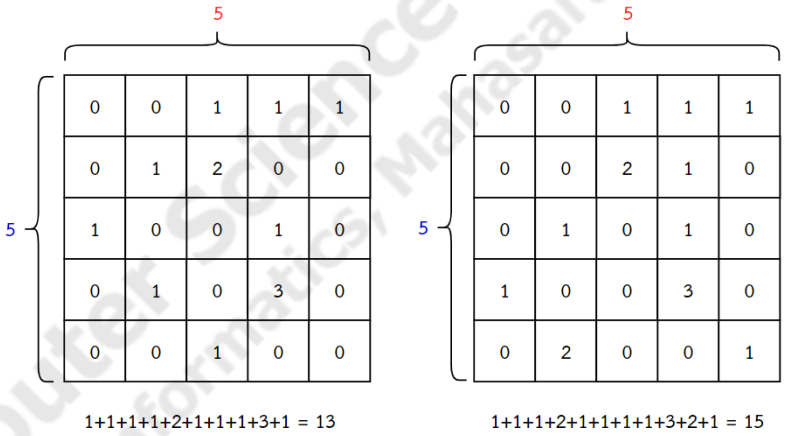
การประเมินคุณภาพของผลลัพธ์ใช้หลักการนำภาพใบหน้าจริงและภาพที่ถูกสร้างขึ้น มาทำการเปรียบเทียบความเหมือนของโครงสร้าง (Structural Similarity Index, SSIM) [13] [14] ซึ่งเป็นวิธีการที่ใช้ในการวัดประสิทธิภาพในด้านเทคนิคของภาพขาว-ดำ (Grayscale) โดยทำการวัดเรื่องของแสงและเงา ความคมชัด โครงสร้าง และความคล้ายคลึงของภาพ ซึ่งจะได้ผลลัพธ์ออกมาเป็นค่าความน่าจะเป็นที่อยู่ในช่วง [0-1]

$$MSE = \frac{1}{m\,n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2 \tag{3.12}$$

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.13}$$

ในการประเมินคุณภาพผลลัพธ์นั้นจำเป็นที่จะต้องใช้ภาพแบบขาว-ดำ จึงต้องทำการแปลงรูปแบบของภาพจากภาพสี (RGB) ให้เป็นภาพขาว-ดำ ก่อนที่จะทำการคำนวณผลลัพธ์ เหตุผลในการใช้ SSIM เนื่องจากผลลัพธ์ที่ได้จากการคำนวณมีค่าที่ไม่แกว่งมากเมื่อเทียบกับการคำนวณความเหมือนของโครงสร้างด้วยเทคนิค Mean square error (MSE) [13] [15] ซึ่งสามารถดูผลลัพธ์ได้ดังภาพประกอบที่ 3.26 โดยสมการคำนวณของ MSE เป็นไปตามสมการ (3.12) และการคำนวณ SSIM เป็นไปตามสมการ (3.13)


**ภาพประกอบที่ 3.26** เปรียนเทียบผลลัพธ์ของการคำนวณแบบ MSE และ SSIM




**ภาพประกอบที่ 3.27** ตัวอย่างการคำนวณผลรวมของภาพในการคำนวณ MSE

การคำนวณ MSE คือการหาค่า Loss ระหว่างทั้งสองรูป โดยการหาผลรวมของค่าทุกพิกเซลในภาพแรก มาลบกับผลรวมของค่าทุกพิกเซลในภาพที่สอง แล้วนำมายกกำลังสอง จากนั้นจะทำการหารด้วยผลรวมทั้งหมดของจำนวนพิกเซล ภาพประกอบที่ 3.27 เป็นการแสดงตัวอย่างการคำนวณผลรวมเพื่อใช้ในการคำนวณ MSE ซึ่งจะได้ผลลัพธ์ดังนี้

$$MSE = \frac{1}{5 \cdot 5} \cdot (13 + 15)$$

$$MSE = \frac{28}{25}$$

$$MSE = 0.0357$$

ตัวอย่างการคำนวณ SSIM โดยกำหนดให้ค่าเฉลี่ย ($\mu$) และส่วนเบี่ยงเบนมาตรฐาน ($\sigma$) ของทั้งสองภาพเป็น [1.0, 0.0] และ [0.7,0.2] ($c_1$ และ $c_2$ คือค่าคงที่ ทำหน้าที่หลีกเลี่ยงความไม่เสถียรในกรณีที่ตัวส่วนมีค่าเข้าใกล้ 0 ซึ่งในที่นี้แทนค่าด้วย 1 ซึ่งสามารถแทนค่าสมการ (3.13) ได้ดังนี้

$$SSIM(x,y) = \frac{(2(1.0)(0.7)+1)\ (2(0.0)+1)}{((1.0)^2+(0.7)^2+1)\ ((0.0)^2+(0.2)^2+1)}$$

$$SSIM(x,y) = \frac{(1.4)\ (1)}{(1.49)\ (1.04)}$$

$$SSIM(x,y) = \frac{1.4}{1.5496}$$

$$SSIM(x,y) = 0.9034$$

จากตัวอย่างผลลัพธ์การคำนวณของทั้งสองวิธี แสดงให้เห็นว่าผลลัพธ์จากการคำนวณด้วย SSIM มีความเหมาะสมในการใช้ตรวจสอบคุณภาพของภาพที่ถูกสร้างขึ้นได้ดีกว่าการคำนวณแบบ MSE ดังนั้นผู้จัดทำจึงได้เลือกวิธีการตรวจสอบคุณภาพผลลัพธ์ด้วยวิธีการ SSIM

3.4.3 การประเมินประสิทธิภาพของแอปพลิเคชัน

การประเมนประสิทธิภาพแอปพลิเคชัน จะทำการประเมินระยะเวลาในการรับ-ส่งข้อมูลระหว่างแอปพลิเคชันและเซิร์ฟเวอร์ ทั้งในส่วนของแอปพลิเคชันบนอุปกรณ์เคลื่อนที่และบนเว็บแอปพลิเคชัน เพื่อวัดประสิทธิภาพการทำงานของแอปพลิเคชัน