Live Facial Recognition

โดย

65010195 นายชลศักดิ์ อนุวารีพงษ์

65010491 นายธีรเมต ช่วยพยุง

65010395 นายธนกฤต ใจประสงค์

65010539 นายนิชรัตน์ ทัดเทียม

โครงงานนี้เป็นส่วนหนึ่งของการศึกษา

วิชา ELEMENTARY DIFFERENTIAL EQUATIONS AND LINEAR ALGEBRA

ปีการศึกษา 2566

ชื่อโครงงาน	Live Facial Recognition (การจดจำใบหน้าแบบ Live)
ผู้จัดทำ	นายชลศักดิ์ อนุวารีพงษ์
	นายธีรเมต ช่วยพยุง
	นายธนกฤต ใจประสงค์
	นายนิชรัตน์ ทัดเทียม
อาจารย์ที่ปรึกษา	อาจารย์ อรฉัตร จิตต์โสภักตร์
ปีการศึกษา	2566

บทคัดย่อ

ในงานวิจัยเรื่อง "Live Facial Recognition" นี้
ผู้วิจัยนำเสนอการใช้เทคโนโลยีการระบุใบหน้าแบบสดในระหว่างการถ่ายทอดภาพ (Live Facial Recognition) โดยใช้อัลกอริทึม MTCNN (Multi-task Cascaded Convolutional Networks) และ VGG16 (Visual Geometry Group 16)
ในกระบวนการตรวจจับและระบุใบหน้า การวิจัยนี้ใช้ทฤษฎีแบบ Linear Algebra
เพื่อทำหน้าที่สำคัญในกระบวนการการระบุใบหน้า อาทิเช่นการแปลงภาพผ่าน Convolution Matrix, Matrix Reshape, และ Matrix Transformation เพื่อให้สามารถนำข้อมูลภาพมาวิเคราะห์ได้อย่างเหมาะสมกับโครงสร้างของโครงข่ายประสาทเทียม (Neural Network) ที่ใช้ในการระบุใบหน้า ผ่านกระบวนการนี้, ระบบสามารถค้นหาความคล้ายหรือค่าความคล้าย (Cosine Similarity) ระหว่างลักษณะที่ได้จากใบหน้าที่ถ่ายทอดมากับฐานข้อมูลใบหน้า
และจัดอันดับลำดับความเป็นไปได้ของการตรวจจับใบหน้าที่สอดคล้องกับคุณสมบัติ การวิจัยนี้นำเสนอแนวทางการพัฒนา Live Facial Recognition ด้วยการประยุกต์ใช้ MTCNN, VGG16, และทฤษฎี Linear Algebra
เพื่อเพิ่มประสิทธิภาพและความแม่นยำในการระบุใบหน้าในระหว่างการถ่ายทอดภาพสด

คำสำคัญ: Live Facial Recognition, MTCNN, VGG16,

บทที่ 1
บทนำ

1. ที่มาและความสำคัญ

การระบุใบหน้าแบบ Live (Live Facial Recognition) เป็นเทคโนโลยีที่กำลังพัฒนาอย่างรวดเร็วในปัจจุบัน การระบุใบหน้าเป็นกระบวนการที่มีการประยุกต์ใช้ในหลายด้านของชีวิตประจำวัน เช่นในระบบควบคุมการเข้าถึงอาคาร, การรักษาความปลอดภัย, การจดจำผู้ใช้ในอุปกรณ์อิเล็กทรอนิกส์, และอีกมากมาย

ทางเราจึงเน้นการวิจัยและพัฒนาทางด้าน Live Facial Recognition โดยใช้อัลกอริทึม MTCNN (Multi-task Cascaded Convolutional Networks) และ VGG16 (Visual Geometry Group 16)
เพื่อการตรวจจับและระบุใบหน้าในระหว่างการถ่ายทอดภาพสด
การระบุใบหน้าในเวลาเจ็ดสามสำคัญอย่างยิ่งในสถานการณ์ที่ต้องการการตอบสนองในเวลาเจ่งสาม
เช่นการควบคุมการเข้าถึงห้องปฏิบัติการสำคัญหรือเต็มไปด้วยการควบคุมความปลอดภัยโดยการใช้อัลกอริทึม MTCNN และ VGG16 ที่เป็นเทคโนโลยีทันสมัยและมีความแม่นยำในการระบุใบหน้า
ซึ่งมีสิ่งสำคัญในงานที่เกี่ยวข้องกับควบคุมการเข้าถึงและความปลอดภัยในเวลา Live
เป็นสิ่งสำคัญในการตรวจจับและระบุใบหน้าขณะที่ข้อมูลถูกถ่ายทอดอยู่
ซึ่งสำคัญในการควบคุมสถานการณ์ในเวลาที่มีความเร่งด่วนโดยใช้ ทฤษฎี Linear Algebra
เป็นส่วนสำคัญในกระบวนการปรับปรุงและความแม่นยำในการระบุใบหน้าและความคล้ายของใบหน้า
ซึ่งทำให้มีความสำคัญในเข้าใจว่าแนวทางการใช้ทฤษฎีนี้มีประโยชน์และความสำคัญต่อการพัฒนาเทคโนโลยี

2. วัตถุประสงค์

2.1 สามารถระบุใบหน้าในระหว่างถ่ายทอดภาพสดได้

2.2 สามารถใช้อัลกอริทึม MTCNN และ VGG16 ในการพัฒนาได้

2.3 สามารถปรับปรุงและระบุด้วยทฤษฎี Linear Algebra

2.4 สามาถระบุใบหน้าในเวลาจริงในเวลานั้นๆได้

2.5 สามารถสร้างโปรแกรมตรวจสอบใบหน้าได้อย่างดีเยี่ยม

3. ผลลัพธ์ที่คาดหวังจากโครงงาน

3.1 การใช้อัลกอริทึมและเทคโนโลยีในสถานการณ์จริง

3.2 การใช้ความรู้ในวิชา ELEMENTARY DIFFERENTIAL EQUATIONS AND LINEAR ALGEBRA นำมาประยุกต์เพื่อใช้เพื่อพัฒนาโครงงานที่ใช้หลักการดังกล่าวเป็นหลัง

บทที่ 2

ภาพรวมการออกแบบระบบ

2.1 ภาพรวมขั้นตอนการทำงานของระบบ

ระบบวิเคราะห์ข้อมูลที่นำเข้าโดยการใช้การเขียนด้วยโปรแกรมภาษา Python

2.1.1 การนำข้อมูลเข้าสู่ระบบ

สร้าง dataset ขึ้นเอง แล้วนำมาเก็บไว้ในโฟลเดอร์ RawData ที่ประกอบด้วยโฟลเดอร์ Test และ Train

2.1.2 การเตรียมข้อมูล

สร้าง dataset ขึ้นเองจากการบันทึกภาพจากกล้องและมีการเก็บ dataset เพิ่มจากการทำ data augmentation (matrix transformation)

2.1.3 การปรับแต่งข้อมูล

การทำ data preprocessing นำ train_set และ validation_set มาทำ data augmentation จากนั้นนำไป train กับ model ที่เตรียมไว้

2.1.4 การวิเคราะห์ข้อมูล

นำค่า loss และ accuracy ของ train set และ validation set มา plot ด้วย matplotlib.pyplot

2.2 รายละเอียดข้อมูลที่เกี่ยวข้อง

สร้าง dataset ขึ้นเองจากการบันทึกรูปภาพจาก face detection แล้วนำรูปภาพไป resize (224,244,3)

```python
import cv2
import numpy as np
from mtcnn import MTCNN

detector = MTCNN()
```

```python
def save_data():
    cap = cv2.VideoCapture(0)
    count = 1
    while True:
        ret, frame = cap.read()
        if ret == False:
            continue
        faces = detector.detect_faces(frame)
        if len(faces) == 0:
            continue
        face = faces[0]
        x, y, w, h = face['box']
        cv2.imshow("frame", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        if cv2.waitKey(1) & 0xFF == ord('s'):
            count += 1
            face_img = frame[y:y+h, x:x+w]
            face_img = cv2.resize(face_img, (224, 224), interpolation=cv2.INTER_AREA)
            cv2.imwrite(f"images/class-3/{count}.jpg", face_img)
            print("Image saved")
    cap.release()
    cv2.destroyAllWindows()
```

สร้าง model แต่งเติมจาก VGG-16

```python
# Load the VGG16 model pretrained on ImageNet
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

```python
# freeze the layers in the base model
for layer in base_model.layers:
    layer.trainable = False
```

```python
# Add a custom classifier
x = base_model.output
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(512, activation='relu')(x)
predictions = Dense(4, activation='softmax')(x)
```

```python
# create the final model
model = Model(inputs=base_model.input, outputs=predictions)
```

```python
# compile the model
model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.001), metrics=['accuracy'])
```

2.3 อธิบายขั้นตอนย่อยแต่ละขั้น

2.3.1 การปรับแต่งข้อมูล

ก่อนนำ data เข้าไป train ให้กับ model ที่เตรียมไว้ ทำ data preprocessing โดยการ resize to (224,224,3) แล้วทำ normalization เราจะสับเปลี่ยนข้อมูลแล้วนำ y train ไปทำ ---> one-hot encoding (นำไปสร้าง model fit เพื่อ generate model)

```python
def load_data(dir_path, image_width, image_height):
    X = []
    y = []
    label = dict()
    i = 0
    for file in os.listdir(dir_path):
        label[i] = file
        for image in os.listdir(dir_path + '/' + file):
            img = cv2.imread(dir_path + '/' + file + '/' + image)
            float_img = img.astype('float32')/255
            resize_img = cv2.resize(float_img, (image_width, image_height))
            X.append(resize_img)
            y.append(i)
        i += 1
    X = np.array(X)
    y = np.array(y)

    return X, y, label
```

```python
# Load the data
X_train, y_train, label = load_data(train_path, 224, 224)
# Shuffle data
s = np.arange(X_train.shape[0])
np.random.shuffle(s)
X_train = X_train[s]
y_train = y_train[s]
# One hot encoding
y_train = to_categorical(y_train, 4)
```

\ จากนั้นนำ data เข้าไปให้ model เพื่อทำการ train

```python
# Train model
hist = model.fit(X_train, y_train, batch_size=20, epochs=10, validation_split=0.3)
```

```
Epoch 1/10
7/7 [==============================] - 30s 4s/step - loss: 7.6982 - accuracy: 0.3357 - val_loss: 2.6167 - val_accuracy: 0.3833
Epoch 2/10
7/7 [==============================] - 30s 4s/step - loss: 1.5183 - accuracy: 0.6500 - val_loss: 1.3568 - val_accuracy: 0.6833
Epoch 3/10
7/7 [==============================] - 30s 4s/step - loss: 0.5679 - accuracy: 0.7929 - val_loss: 0.6932 - val_accuracy: 0.7333
Epoch 4/10
7/7 [==============================] - 30s 4s/step - loss: 0.2517 - accuracy: 0.8929 - val_loss: 0.1986 - val_accuracy: 0.9667
Epoch 5/10
7/7 [==============================] - 30s 5s/step - loss: 0.1650 - accuracy: 0.9357 - val_loss: 0.0772 - val_accuracy: 0.9667
Epoch 6/10
7/7 [==============================] - 29s 4s/step - loss: 0.0538 - accuracy: 0.9714 - val_loss: 0.1542 - val_accuracy: 0.9333
Epoch 7/10
7/7 [==============================] - 30s 4s/step - loss: 0.0254 - accuracy: 0.9929 - val_loss: 0.0340 - val_accuracy: 0.9833
Epoch 8/10
7/7 [==============================] - 30s 4s/step - loss: 0.0248 - accuracy: 0.9857 - val_loss: 0.0167 - val_accuracy: 1.0000
Epoch 9/10
7/7 [==============================] - 30s 5s/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.0082 - val_accuracy: 1.0000
Epoch 10/10
7/7 [==============================] - 29s 4s/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0070 - val_accuracy: 1.0000
```

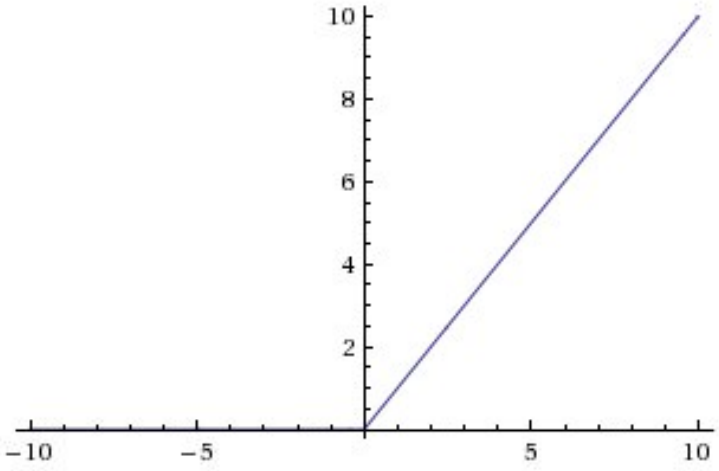
บทที่ 3
การประยุกต์ใช้ทฤษฎี

3.1 การประยุกต์ใช้ทฤษฎีเวกเตอร์

3.1.1 ReLu Activation function (Rectified Linear Unit)

วิธีที่ ReLU จับผลกระทบที่เป็นเส้นตรงและผลกระทบที่ไม่เป็นเส้นตรง

ผลกระทบที่เป็นเส้นตรง: พิจารณาโหนดเดียวในโมเดลเครือข่ายประสาทเทียม (neural network) โดยเริ่มด้วยการสมมติว่ามีอินพุตสองตัวคือ A และ B น้ำหนักจาก A และ B เข้าไปในโหนดเรามีค่าเป็น 2 และ 3 ตามลำดับ ดังนั้นผลลัพธ์ของโหนดคือ f(2A+3B) โดยเราใช้ฟังก์ชัน ReLU สำหรับ f ดังนั้นถ้า 2A+3B เป็นบวก ค่าผลลัพธ์ของโหนดเราก็เป็น 2A+3B หาก 2A+3B เป็นลบ ค่าผลลัพธ์ของโหนดเราก็เป็น 0



สำหรับความชัดเจน พิจารณากรณีที่ A=1 และ B=1 ผลลัพธ์คือ 2A+3B และถ้า A เพิ่มขึ้น ผลลัพธ์ก็เพิ่มขึ้นเช่นกัน อย่างอื่น ถ้า B=-100 ผลลัพธ์คือ 0 และถ้า A เพิ่มขึ้นเล็กน้อย ผลลัพธ์ยังคงเป็น 0 ดังนั้น A อาจเพิ่มผลลัพธ์ของเราหรือไม่เพิ่มขึ้นก็ขึ้นอยู่กับค่าของ B นี่คือกรณีที่โหนดจับผลกระทบที่เป็นเส้นตรง การเพิ่มโหนดและเลเยอร์เพิ่มเติมจะทำให้ความซับซ้อนของผลกระทบเพิ่มขึ้นเท่านั้น แต่คุณควรเห็นว่าฟังก์ชันการกระตุ้นช่วยจับผลกระทบที่เป็นเส้นตรง

ผลกระทบที่ไม่เป็นเส้นตรง: ฟังก์ชันคือไม่เป็นเส้นตรงถ้าความชันไม่คงที่ ดังนั้น ReLU คือฟังก์ชันที่ไม่เป็นเส้นตรงรอบ 0 แต่ความชันเสมอเป็น 0 (สำหรับค่าลบ) หรือ 1 (สำหรับค่าบวก) นี่คือประเภทของความไม่เป็นเส้นตรงที่จำกัดมาก

แต่มีข้อเท็จจริงสองข้อเกี่ยวกับโมเดลการเรียนรู้ลึก (deep learning) ที่ช่วยให้เราสร้างประเภทของความไม่เป็นเส้นตรงจากวิธีที่เรารวมโหนด ReLU กัน

3.1.2 SoftMax Function หรือ Normalized Exponential Function

ฟังก์ชันที่รับ Input เป็น Vector ของ Logit จำนวนจริง แล้ว Normalize ออกมาเป็นความน่าจะเป็น Probability ที่ผลรวมเท่ากับ 1 Softmax มักถูกนำไปไว้ Layer สุดท้าย ของ Neural Network เพื่อให้ Output ออกมาเป็น Probability ไปคำนวน Negative Log Likelihood เป็น Cross Entropy Loss เช่น ในงาน Single Class Classification Softmax ถูกนำไปใช้บ่อย ในงาน Classification จนถึงขนาดมีคนเรียกว่า Softmax Classifier หรือ Softmax Loss

$$\sigma(\mathbf{z})i = \frac{e^{z_i}}{\sum j=1^K e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in R^K$$

3.1.3 accuracy score

ค่าความถูกต้องของข้อมูล หาได้จาก ค่าที่โมเดลทายถูกทั้งหมด/ค่าทั้งหมด

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

3.1.4 confusion matrix

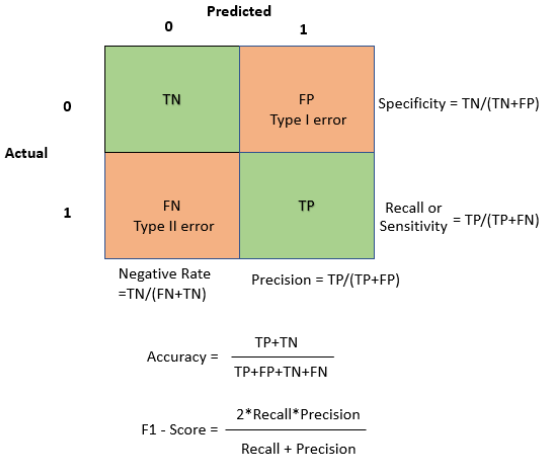
การเปรียบเทียบอัตราส่วนระหว่างสิ่งที่ทำนายกับสิ่งที่เกิดขึ้นจริง

True Positive (TP)= สิ่งที่ทำนาย ตรงกับสิ่งที่เกิดขึ้นจริง ในกรณี ทำนายว่าจริง และสิ่งที่เกิดขึ้น ก็คือ จริง

True Negative (TN)= สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น ในกรณี ทำนายว่า ไม่จริง และสิ่งที่เกิดขึ้น ก็คือ ไม่จริง

False Positive (FP)= สิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น คือทำนายว่า จริง แต่สิ่งที่เกิดขึ้น คือ ไม่จริง

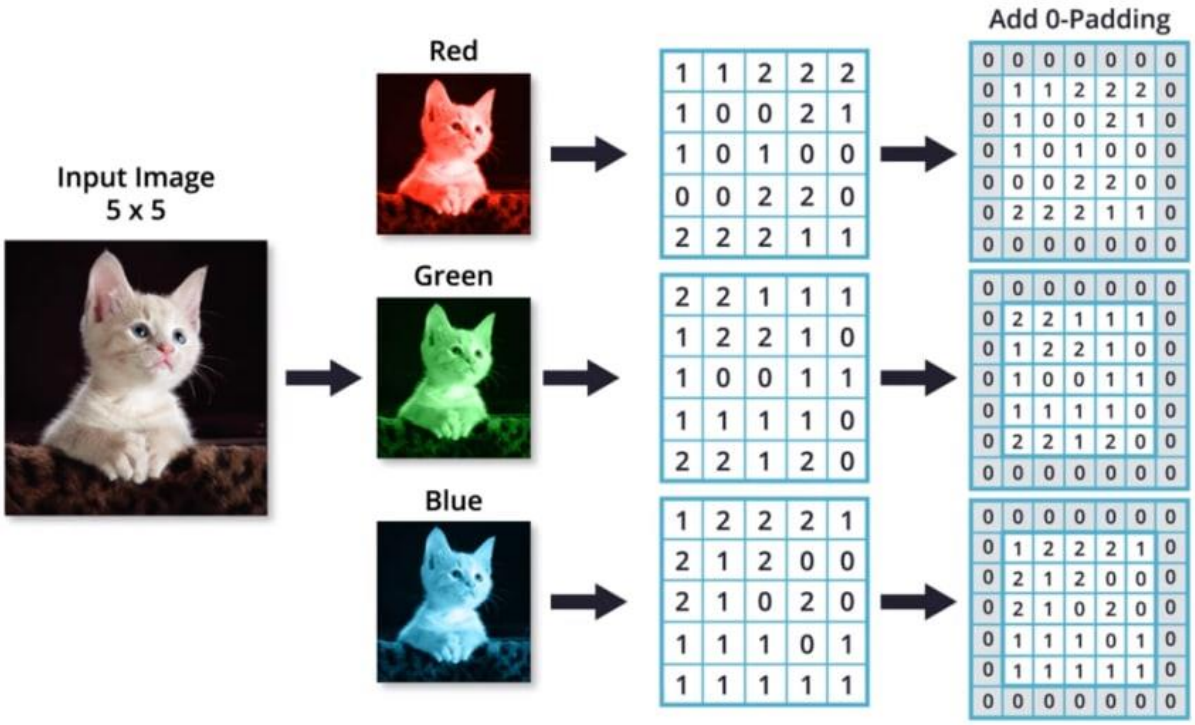
False Negative (FN)= สิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง คือทำนายว่าไม่จริง แต่สิ่งที่เกิดขึ้น คือ จริง



3.2 การประยุกต์ใช้ทฤษฎีเมทริกซ์

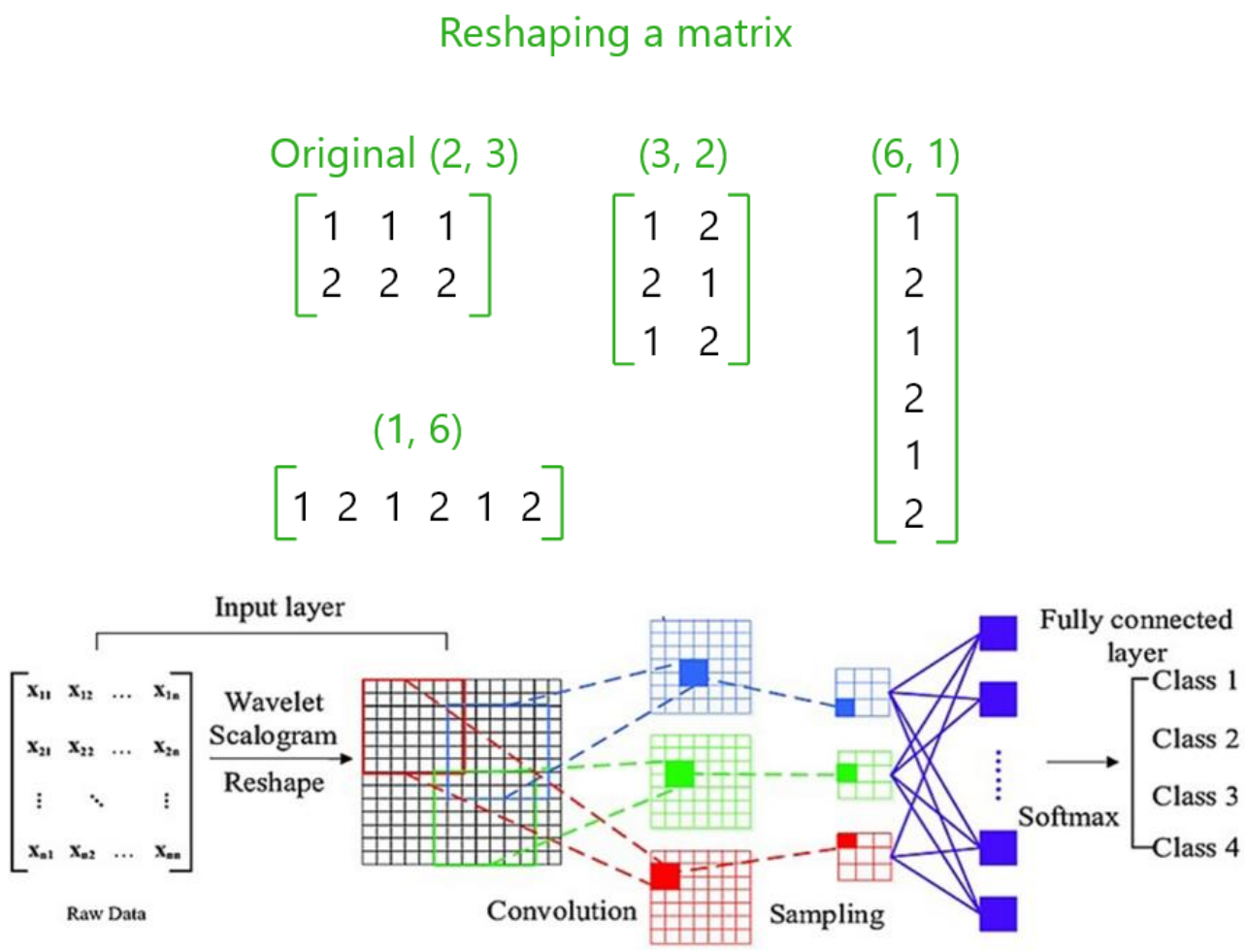
3.2.1 Convolution Matrix

เป็นการ dot product ของ input กับตัว filter เพื่อนำข้อมูลมาวิเคราห์ ในที่นี้จะจัดการกับ input ภาพสีในระบบ RGB แล้วส่งไปที่ Neural Network Layer ถัดไป



3.2.2 Matrix Reshape

เป็นการนำสมาชิกของ array มาจัดเรียงใน dimension ใหม่ตามที่เราต้องการ โดยจะยังคงข้อมูลใน array ไว้เหมือนเดิม



เนื่องจากการนำข้อมูลเข้า Neural networks แต่ละข้อมูลต้องเป็น format เดียวกัน จึงต้องทำการ reshape ก่อนเสมอ

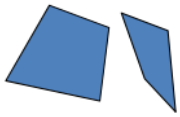
3.2.3 Matrix Transformation

เป็นการนำข้อมูลมาปรับรูปแบบในระบบพิกัดฉาก เช่นการ rotation, scaling, translation เนื่องจากการตรวจจับใบหน้า ตำแหน่งใบหน้าของ user มีโอกาสที่จะวางไม่ตรงกับข้อมูลที่เราเก็บไว้ เพื่อลดการเกิด error จึงใช้ร่วมกับอัลกอริทึมของการตรวจจับใบหน้า เพื่อมีประสิทธิภาพที่เพิ่มขึ้น

A square transforms to:

Projective 8dof

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

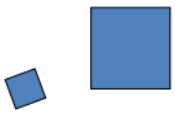


Affine 6dof

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

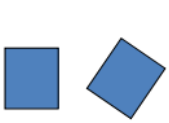
Similarity 4dof

$$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

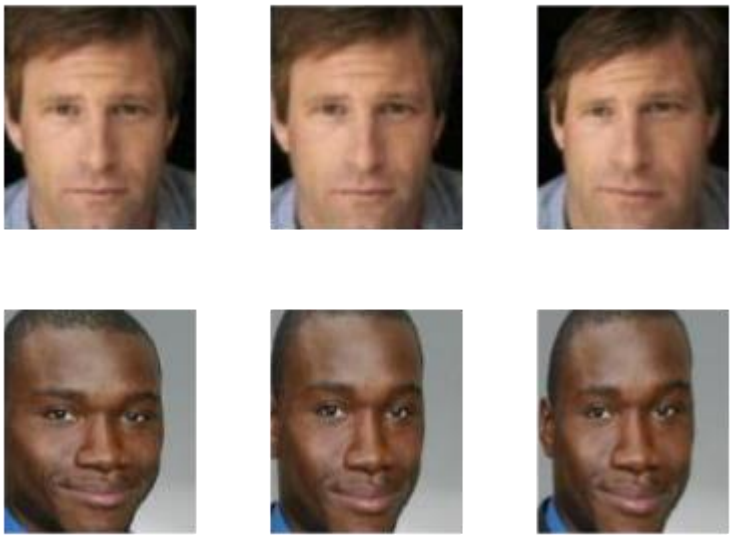


Euclidean 3dof

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Similarity Affine Projective

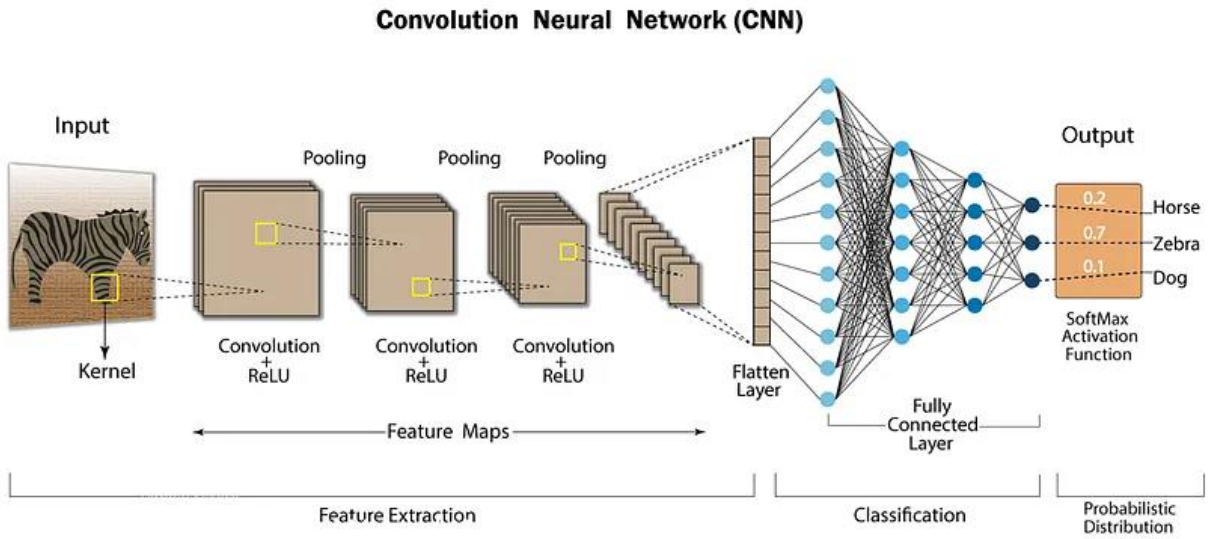


3.3 การประยุกต์ใช้ Neural networks

คือโมเดลในการประมวลผลข้อมูลรูปแบบหนึ่ง โดยจะแบ่งเป็นnode แต่ละnodeจะรับค่าตัวแปร คูณกับค่าน้ำหนัก รวม และส่งต่อไปยังnodeถัดไป เพื่อที่จะออกข้อสรุปไปยัง output ดังภาพตัวอย่าง

พอประมวลผลภาพแล้วจะมีน้ำหนักว่าเป็นสัตว์ชนิดใด ในที่นี้จะใช้ในการตรวจจับใบหน้า โดยใช้ MTCNN

MTCNN (Multi-Task Cascaded Convolutional Neural Networks) คือ โมเดลการตรวจจับใบหน้า ซึ่งมีประสิทธิภาพสูง เนื่องจากสามารถ ตรวจจับใบหน้า(Face Detection) และจัดตำแหน่งใบหน้า(Face Alignment) ในเวลาเดียวกันได้



บทที่ 4

ผลการทดลอง

จากขั้นตอนการทำงานทั้งหมด การนำเข้าข้อมูลจนถึงขั้นตอน Evaluation เพื่อสร้างชุดข้อมูลในการประมวลผลการ predict จาก model มีการการทำนายผลดังนี้

4.1 Matrix Transformation

Original image

Transformation Image

จากผลการทดลอง จะเห็นว่าทาง dataset ของ face recognition มีจำนวนที่ถ่ายภาพได้ยังไม่ครอบคลุม จึงทำการ Transformation ด้วยการ rotate หรือการ resize เพื่อให้เกิด dataset จำนวนใหม่ขึ้นมา

4.2 Convolution

```
face_recog_model.summary()
```

[27] ✓ 0.1s

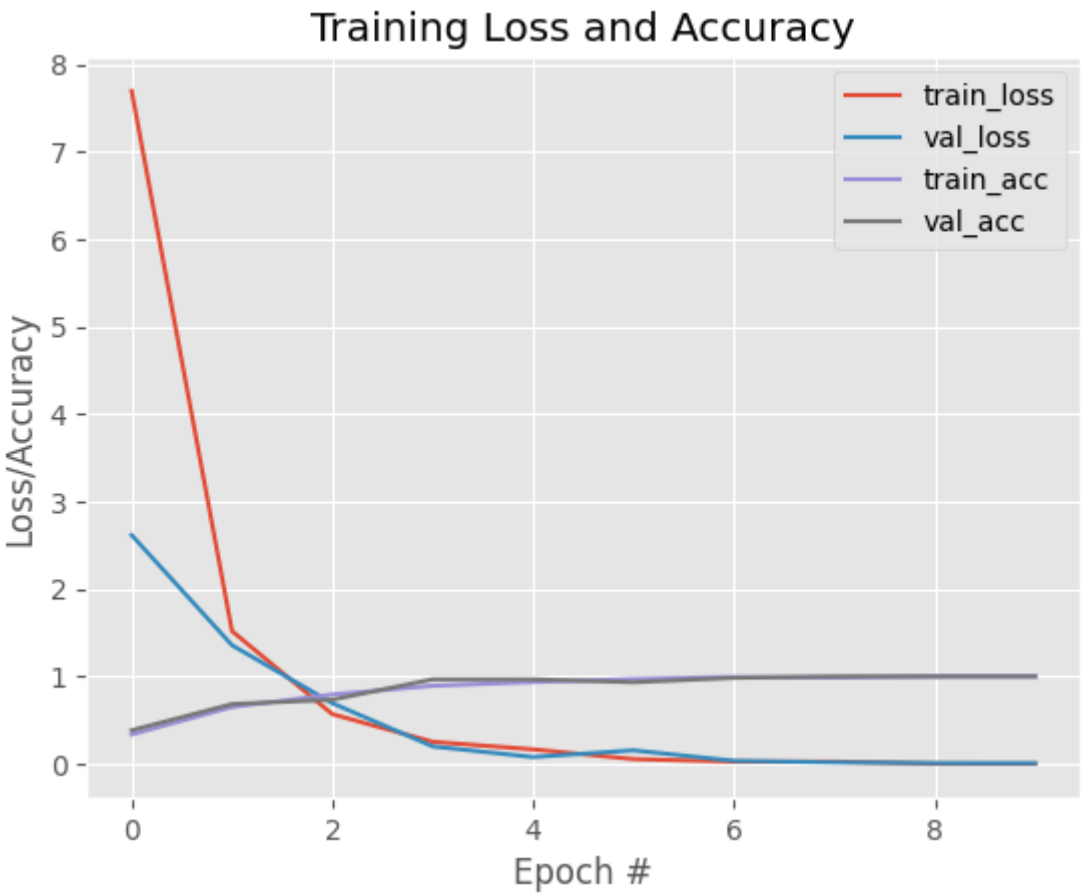
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
...
Total params: 67220228 (256.42 MB)
Trainable params: 52505540 (200.29 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

จากผลการทดลอง จะเห็นว่าเราได้ทำการรับ Input เป็น image shape 224x224 จากนั้นทำการ reshape ด้วย convolution เพื่อนำข้อมูลไป flatten นำข้อมูลไปใช้ต่อไปในขนาด 28x28

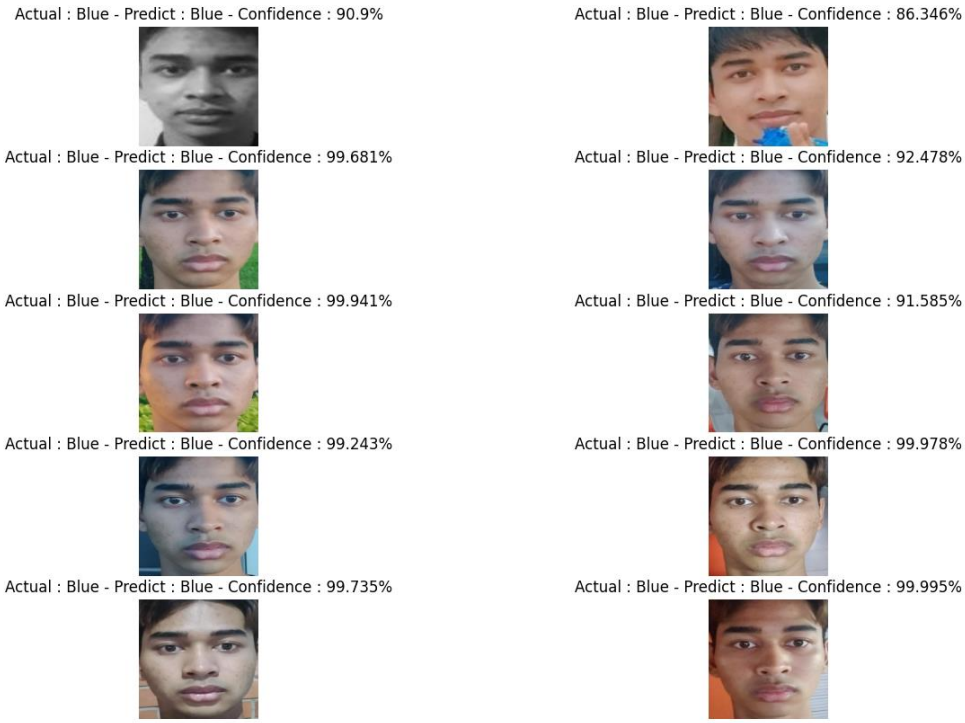
4.3 Evaluation / Accuration

```
# Train model
hist = model.fit(X_train, y_train, batch_size=20, epochs=10, validation_split=0.3)
```

```
Epoch 1/10
7/7 [==============================] - 30s 4s/step - loss: 7.6982 - accuracy: 0.3357 - val_loss: 2.6167 - val_accuracy: 0.3833
Epoch 2/10
7/7 [==============================] - 30s 4s/step - loss: 1.5183 - accuracy: 0.6500 - val_loss: 1.3568 - val_accuracy: 0.6833
Epoch 3/10
7/7 [==============================] - 30s 4s/step - loss: 0.5679 - accuracy: 0.7929 - val_loss: 0.6932 - val_accuracy: 0.7333
Epoch 4/10
7/7 [==============================] - 30s 4s/step - loss: 0.2517 - accuracy: 0.8929 - val_loss: 0.1986 - val_accuracy: 0.9667
Epoch 5/10
7/7 [==============================] - 30s 5s/step - loss: 0.1650 - accuracy: 0.9357 - val_loss: 0.0772 - val_accuracy: 0.9667
Epoch 6/10
7/7 [==============================] - 29s 4s/step - loss: 0.0538 - accuracy: 0.9714 - val_loss: 0.1542 - val_accuracy: 0.9333
Epoch 7/10
7/7 [==============================] - 30s 4s/step - loss: 0.0254 - accuracy: 0.9929 - val_loss: 0.0340 - val_accuracy: 0.9833
Epoch 8/10
7/7 [==============================] - 30s 4s/step - loss: 0.0248 - accuracy: 0.9857 - val_loss: 0.0167 - val_accuracy: 1.0000
Epoch 9/10
7/7 [==============================] - 30s 5s/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 0.0082 - val_accuracy: 1.0000
Epoch 10/10
7/7 [==============================] - 29s 4s/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0070 - val_accuracy: 1.0000
```

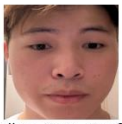


Evaluation (การประเมินผล)



Actual : New - Predict : New - Confidence : 51.489%

Actual : New - Predict : New - Confidence : 99.198%



Actual : New - Predict : New - Confidence : 97.976%

Actual : New - Predict : New - Confidence : 46.627%

Actual : New - Predict : New - Confidence : 37.562%

Actual : New - Predict : New - Confidence : 99.76%

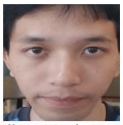
Actual : New - Predict : New - Confidence : 99.996%

Actual : New - Predict : New - Confidence : 99.002%

Actual : New - Predict : New - Confidence : 99.4%

Actual : New - Predict : New - Confidence : 99.101%

Actual : Pond - Predict : Pond - Confidence : 100.0%



Actual : Pond - Predict : Pond - Confidence : 99.689%

Actual : Pond - Predict : Pond - Confidence : 99.749%

Actual : Pond - Predict : Pond - Confidence : 99.981%

Actual : Pond - Predict : Pond - Confidence : 99.993%

Actual : Pond - Predict : Pond - Confidence : 99.999%

Actual : Pond - Predict : Pond - Confidence : 99.951%

Actual : Pond - Predict : Pond - Confidence : 100.0%

Actual : Pond - Predict : Pond - Confidence : 99.577%

Actual : Pond - Predict : Pond - Confidence : 99.89%

Actual : Tar - Predict : Tar - Confidence : 96.47%

Actual : Tar - Predict : Tar - Confidence : 99.284%

Actual : Tar - Predict : Tar - Confidence : 99.777%

Actual : Tar - Predict : Tar - Confidence : 99.875%

Actual : Tar - Predict : Tar - Confidence : 99.751%

Actual : Tar - Predict : Tar - Confidence : 97.757%

Actual : Tar - Predict : Tar - Confidence : 96.728%

Actual : Tar - Predict : Tar - Confidence : 99.015%

Actual : Tar - Predict : Tar - Confidence : 98.008%

Actual : Tar - Predict : Tar - Confidence : 99.276%

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 การสรุปผลการทดลอง

จากการศึกษาโครงงานเรื่อง face recognition จากการใช้ ความรู้มาทำการวิจัยมุ่งเน้นการระบุใบหน้าแบบสด (Live Facial Recognition) โดยนำเสนอการใช้เทคโนโลยี MTCNN และ VGG16 ในกระบวนการตรวจจับและระบุใบหน้าในเวลาที่กำลังถ่ายทอดภาพสด โดยเฉพาะทฤษฎี Linear Algebra ได้ถูกนำมาใช้เพื่อปรับปรุงระบบที่เกี่ยวข้อง เช่นการแปลงภาพผ่าน Convolution Matrix, Matrix Reshape, และ Matrix Transformation เพื่อให้ข้อมูลภาพเข้ากับโครงสร้างของระบบประสาทเทียมได้อย่างเหมาะสม

การนำข้อมูลที่ได้จากการระบุใบหน้ามาวิเคราะห์และนำมาเทียบเท่า (Cosine Similarity) กับฐานข้อมูลใบหน้าเพื่อจัดอันดับความเป็นไปได้ของการตรวจจับใบหน้าที่เหมาะสมกับคุณสมบัติ เป็นสิ่งสำคัญในการพัฒนาระบบระบุใบหน้าในสถานการณ์การถ่ายทอดภาพสด

ผลการทดลองแสดงให้เห็นว่าระบบสามารถทำงานอย่างมีประสิทธิภาพและความแม่นยำในการระบุใบหน้าในระหว่างการถ่ายทอดภาพสดได้อย่างสำเร็จ

5.2 ข้อเสนอแนะ

5.2.1 ปัญหาที่พบ

- Dataset น้อยทำให้เวลา Train model จะทำให้ค่า accuracy ที่สูง
- การ predict Real Time ผลพวงจาก dataset และ model VGG16 ทำให้ predict recognize หลุดไปเป็นคนอื่นได้ ในบางช่วงถ้าไม่ชัดเจนใน data image input พอ

5.2.2 ข้อเสนอแนะ

- ควรจัดหา Dataset สำหรับคนที่ไม่มี classification ให้มากขึ้นกว่าเดิม แล้วตัดเกณฑ์ที่ unknown ในกรณีที่ไม่มีข้อมูลของคนที่มี data face recognition

บรรณานุกรม

Face Recognition — ระบบตรวจสอบใบหน้า [TH] https://medium.com/@chonlawit.kingz/face-recognition-%E0%B8%A3%E0%B8%B0%E0%B8%9A%E0%B8%9A%E0%B8%95%E0%B8%A3%E0%B8%A7%E0%B8%88%E0%B8%AA%E0%B8%AD%E0%B8%9A%E0%B9%83%E0%B8%9A%E0%B8%AB%E0%B8%99%E0%B9%89%E0%B8%B2-th-d787e0d55f1f

Philipp Wagner, Face Recognition with Python facerec_python.pdf (bytefish.de)

Understanding the Convolutional Filter Operation in CNN's | Advanced Deep Learning (medium.com)

Machine Learning - Convolution with color images - DEV Community

Face Detection using MTCNN — a guide for face extraction with a focus on speed | by Justin Güse | Towards Data Science

Reshaping the Dataset For Neural Networks | by Abhishek Shah | Medium

Demystifying Face Recognition IV: Face-Alignment (melgor.github.io)

Convolutional Neural Networks for Dummies | by Prathammodi | Oct, 2023 | Medium