# Facial Emotion Recognition
# using Convolutional Neural Network (CNN)


**จัดทำโดย**

1.นางสาวนภัสนันท์ ดามะพร รหัสนิสิต 6610502102

2.นายสรัล สังข์วร รหัสนิสิต 6610505594


**เสนอ**

ผศ.ดร.ภารุจ รัตนวรพันธุ์

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา

การเรียนรู้เชิงลึก (Deep Learning) 01204466-65

ภาคเรียนที่ 1 ปีการศึกษา 2568

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตบางเขน

## 1.หัวข้อนี้น่าสนใจอย่างไร ทำไมถึงเลือกหัวข้อนี้

ความน่าสนใจของหัวข้อ Facial Emotion Recognition เกิดจากการที่ "อารมณ์" เป็นส่วนสำคัญของการสื่อสารระหว่างมนุษย์ และมีผลโดยตรงต่อพฤติกรรมและการตัดสินใจในสถานการณ์ต่าง ๆ หากคอมพิวเตอร์สามารถจดจำอารมณ์ของผู้ใช้งานได้ ก็จะช่วยให้ระบบสามารถโต้ตอบได้อย่างเหมาะสมและมีประสิทธิภาพมากขึ้น ซึ่งสอดคล้องกับแนวคิดของ Human-Computer Interaction ที่กำลังได้รับความสนใจอย่างกว้างขวางในปัจจุบัน

อีกทั้งการจำแนกอารมณ์จากใบหน้ายังเป็นปัญหาที่ท้าทาย เนื่องจากลักษณะใบหน้าของคนแต่ละคนแตกต่างกัน รวมถึงมีตัวแปรอื่น ๆ เช่น แสง มุมกล้อง และความเข้มของอารมณ์ ทำให้เป็นปัญหาที่เหมาะสมกับการประยุกต์ใช้เทคนิคด้าน Deep Learning เพื่อช่วยในการเรียนรู้ลักษณะรูปแบบเชิงภาพที่ซับซ้อน

**เหตุผลที่เลือกหัวข้อนี้**

1. **ความท้าทายเชิงเทคนิค**
   การจำแนกอารมณ์เป็นปัญหาที่ต้องวิเคราะห์รายละเอียดของสีหน้าในระดับเล็กมาก เช่น การขยับคิ้ว ดวงตา หรือรูปปาก ซึ่งมีความแตกต่างกันเพียงเล็กน้อยระหว่างอารมณ์
2. **ความเป็นไปได้ในการประยุกต์ใช้งานจริง**
   ระบบนี้สามารถนำไปใช้งานได้ในหลายด้าน เช่น
   - ระบบช่วยการเรียนรู้ที่ปรับเนื้อหาตามอารมณ์ผู้เรียน
   - ระบบบริการลูกค้าหรือ Call Center ที่ตอบสนองผู้ใช้ตามอารมณ์
   - การติดตามสุขภาพจิต เช่น ผู้ป่วยซึมเศร้า

## 2. ทำไมหัวข้อนี้จึงต้องใช้ Deep Learning

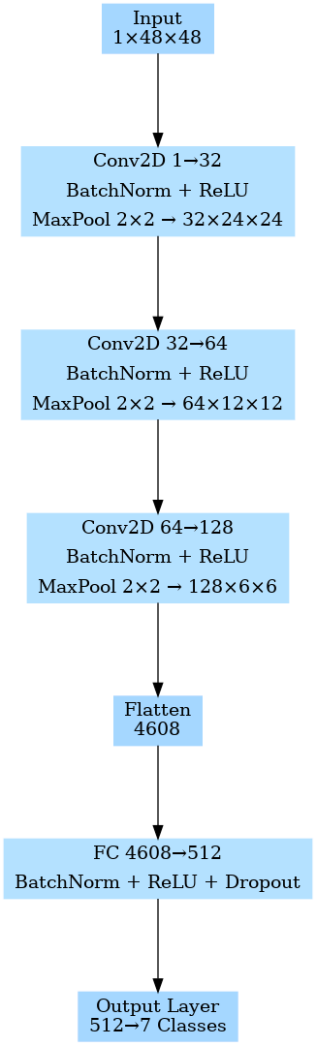
การจำแนกอารมณ์จากใบหน้าเป็นปัญหาที่มีความซับซ้อนสูง เนื่องจากลักษณะของอารมณ์ไม่ได้ถูกกำหนดจากเพียงตำแหน่งของอวัยวะบนใบหน้าเท่านั้น แต่ยังรวมถึงความเข้มของการแสดงออก มุมมอง แสง สีผิว และความแตกต่างเฉพาะบุคคล ทำให้การใช้วิธีการดั้งเดิมในการดึงลักษณะเด่น (Feature Engineering) เช่น HOG, LBP หรือ SVM มีข้อจำกัด และมักให้ผลลัพธ์ไม่สม่ำเสมอเมื่อสภาพแวดล้อมเปลี่ยนแปลงไป

ในขณะที่ Deep Learning โดยเฉพาะ Convolutional Neural Network (CNN) สามารถเรียนรู้รูปแบบของภาพได้โดยอัตโนมัติ โดยไม่ต้องอาศัยการออกแบบคุณลักษณะด้วยมือ จึงเหมาะอย่างยิ่งสำหรับปัญหาด้าน Image Recognition และ Facial Expression Analysis และเหตุผลที่ต้องใช้ Deep learning ได้แก่

1. **ลักษณะของปัญหาเป็นแบบเชิงภาพ (Image-based Problem)**
   อารมณ์ถูกสื่อสารผ่านการเปลี่ยนแปลงเชิงพื้นที่ของใบหน้า เช่น การยิ้ม การขมวดคิ้ว การหรี่ตา ซึ่งเป็นข้อมูลประเภทภาพ CNN ถูกออกแบบมาเพื่อเรียนรู้ลักษณะเชิงพื้นที่โดยตรง จึงตอบโจทย์ปัญหานี้ได้ดีกว่าวิธี Machine Learning แบบดั้งเดิม
2. **CNN สามารถเรียนรู้ Feature ที่ซับซ้อนได้โดยอัตโนมัติ**
   แทนที่มนุษย์จะต้องออกแบบ Feature เอง (ซึ่งอาจไม่ครอบคลุม), CNN สามารถเรียนรู้ตั้งแต่โครงสร้างพื้นฐาน (เช่น ขอบและเส้น) ไปจนถึงลักษณะใบหน้าที่ยากและมีความหมาย (เช่น ความตึงของกล้ามเนื้อรอบปากหรือดวงตา)
3. **มีความยืดหยุ่นและประสิทธิภาพสูงกว่า Traditional ML**
   วิธีแบบ SVM, k-NN, หรือ Random Forest ต้องพึ่ง Feature Extraction ก่อนเสมอ และประสิทธิภาพจะขึ้นกับวิธีออกแบบ Feature ในขณะที่ CNN เรียนรู้ทั้งหมดในขั้นตอนเดียว ทำให้ได้ผลลัพธ์ที่มีเสถียรภาพและแม่นยำมากกว่า

## 3. สถาปัตยกรรมของโมเดล (Model Architecture)

โมเดลที่ใช้ในโครงงานนี้เป็นโมเดลประเภท Convolutional Neural Network (CNN) ซึ่งเหมาะสำหรับงานประมวลผลภาพ เนื่องจากสามารถเรียนรู้ลักษณะเชิงพื้นที่ของภาพ (Spatial Features) ได้โดยอัตโนมัติ โมเดลถูกออกแบบให้มีโครงสร้างแบบเรียงลำดับจากการดึงลักษณะทั่วไปของภาพไปจนถึงลักษณะเชิงอารมณ์ที่ละเอียดขึ้น ก่อนจะเชื่อมเข้าสู่ชั้น Fully Connected เพื่อทำนายผลลัพธ์ในรูปแบบของคลาสอารมณ์ทั้ง 7 ประเภท ได้แก่ Angry, Disgust, Fear, Happy, Sad, Surprise และ Neutral ดังแสดงในภาพด้านล่างนี้



## โครงสร้างของโมเดลประกอบด้วย

| ชั้น (Layer) | รายละเอียด | ขนาดเอาต์พุต |
|---|---|---|
| Input Layer | ภาพขาวดำขนาด 48×48 พิกเซล (1 ช่องสัญญาณ) | 1 × 48 × 48 |
| Convolution Block 1 | Conv2D 1→32 + BatchNorm + ReLU + MaxPool(2×2) | 32 × 24 × 24 |
| Convolution Block 2 | Conv2D 32→64 + BatchNorm + ReLU + MaxPool(2×2) | 64 × 12 × 12 |
| Convolution Block 3 | Conv2D 64→128 + BatchNorm + ReLU + MaxPool(2×2) | 128 × 6 × 6 |
| Flatten Layer | แปลงข้อมูล 3 มิติให้เป็นเวกเตอร์ 1 มิติ | 4608 nodes |
| Fully Connected Layer | Linear 4608→512 + BatchNorm + ReLU + Dropout | 512 |
| Output Layer | Linear 512→7 (จำนวนอารมณ์ทั้งหมด) | 7 Classes |

## คำอธิบายหลักการทำงาน

1. **Convolution Layers**
   ทำหน้าที่ดึงลักษณะพื้นฐานของภาพ เช่น เส้น ขอบ รูปร่างเบื้องต้น ก่อนจะค่อยๆ เรียนรู้ลักษณะที่เฉพาะเจาะจงของอารมณ์มากขึ้นเมื่อชั้นมีจำนวนฟีเจอร์มากขึ้น
2. **Batch Normalization**
   ช่วยให้การเรียนรู้มีความเสถียร ลดการกระจายของค่า Activation และช่วยให้โมเดลเทรนได้ไวขึ้น
3. **ReLU Activation Function**
   ช่วยให้โมเดลสามารถเรียนรู้ความสัมพันธ์แบบไม่เชิงเส้น ซึ่งสำคัญในการจำแนกอารมณ์ที่มีความแตกต่างละเอียด
4. **MaxPooling Layers**
   ลดขนาดข้อมูลลง แต่คงข้อมูลลักษณะเด่น ทำให้โมเดลประหยัดหน่วยความจำและลดโอกาส Overfitting
5. **Fully Connected Layer + Dropout**
   เชื่อมข้อมูลที่สกัดแล้วไปยังขั้นตอนการจำแนก พร้อม Dropout ช่วยป้องกันการจำค่ามากเกินไป
6. **Output Layer**
   ทำนายผลลัพธ์เป็นหนึ่งใน 7 คลาสอารมณ์ตามค่าความน่าจะเป็นที่โมเดลเรียนรู้

## 4. อธิบายโค้ดและโครงสร้างระบบ

โครงงานนี้ใช้ PyTorch เป็น Framework หลักในการสร้างและฝึกสอนโมเดล Convolutional Neural Network (CNN) สำหรับจำแนกอารมณ์จากใบหน้า โดยมีการทำงานแบ่งออกเป็นขั้นตอนดังนี้

### ส่วนที่ 1: การนำเข้า Libraries และการตั้งค่าเบื้องต้น

- นำเข้า Libraries ที่จำเป็น เช่น `torch`, `torch.nn`, `torchvision.transforms`, `numpy`, `pandas`, `matplotlib`, `seaborn`, และ `cv2`
- ตรวจสอบและตั้งค่าอุปกรณ์ประมวลผล (Device) ให้รองรับ GPU หากมีการเชื่อมต่อ เพื่อเพิ่มความเร็วในการฝึกโมเดล
  `device = torch.device("cuda"if torch.cuda.is_available() else "cpu")`

### ส่วนที่ 2: การจัดเตรียมและประมวลผลข้อมูล (Data Processing)

- ใช้ชุดข้อมูล **FER2013** ซึ่งอยู่ในรูป `.csv` ประกอบด้วยคอลัมน์ `emotion`, `pixels`, และ `Usage`
- สร้างคลาส `FER2013Dataset` เพื่ออ่านข้อมูล, แปลง string ของ pixel เป็นภาพ 48×48 และกำหนด label (emotion)
- เตรียม Transform สำหรับการฝึกและทดสอบโมเดล
  - แปลงเป็นภาพ Grayscale
  - ปรับขนาดเป็น 48×48
  - Normalize ค่า pixel ให้อยู่ในช่วง (-1, 1)
- แบ่งข้อมูลเป็น 3 ส่วน ได้แก่
  Training (Training), Validation (PublicTest) และ Testing (PrivateTest)
- ใช้ `DataLoader` ของ PyTorch เพื่อโหลดข้อมูลเป็น Batch ขนาด 64

## ส่วนที่ 3: การสร้างโมเดล (Model Construction)

สร้างคลาส **`SimpleCNN_with_BatchNorm`** สืบทอดจาก `nn.Module`
โมเดลนี้ประกอบด้วย 3 Convolutional Blocks ตามลำดับ พร้อม Batch Normalization และ Dropout

| **Layer** | **ขนาดอินพุต → เอาต์พุต** | **พารามิเตอร์** | **Activation Function** |
|---|---|---|---|
| Conv1 + BN1 + Pool | 1×48×48 → 32×24×24 | (3×3×1×32) + 32 | ReLU |
| Conv2 + BN2 + Pool | 32×24×24 → 64×12×12 | (3×3×32×64) + 64 | ReLU |
| Conv3 + BN3 + Pool | 64×12×12 → 128×6×6 | (3×3×64×128) + 128 | ReLU |
| Fully Connected | 4608 → 512 | (4608×512) + 512 | ReLU + Dropout(0.5) |
| Output Layer | 512 → 7 | (512×7) + 7 | Softmax |

## ส่วนที่ 4: การตั้งค่า Loss Function และ Optimizer

- Loss Function ที่ใช้: **Cross Entropy Loss**
- Optimizer ที่ใช้: **Adam Optimizer** (Learning rate = 0.001)

### ส่วนที่ 5: ขั้นตอนการฝึกโมเดล (Training Process)

- ตั้งค่า Epoch = 50
- ใช้โหมด `model.train()` เพื่อเปิด Dropout และ BatchNorm สำหรับการฝึก
- วนลูปฝึกโมเดล โดยคำนวณ Loss และอัปเดต Weight ด้วย Backpropagation
- หลังจบแต่ละ Epoch จะคำนวณ **Validation Loss และ Accuracy** และเก็บไว้ใน list `train_losses`, `val_losses`, `val_accuracies`
- เมื่อการฝึกจบลง จะบันทึกน้ำหนักโมเดลลงไฟล์ `fer2013_cnn_model_with_evaluate.pth`

### ส่วนที่ 6: การประเมินผล (Model Evaluation)

ในขั้นตอนนี้จะใช้ชุดข้อมูลทดสอบเพื่อวัดประสิทธิภาพของโมเดล
โดยใช้ Metrics ได้แก่ Accuracy, Precision, Recall, F1-score และ Confusion Matrix พร้อมทั้งพล็อตกราฟ Training/Validation Loss และ Accuracy ต่อ Epoch เพื่อแสดงแนวโน้มการเรียนรู้ของโมเดล

### ส่วนที่ 7: การเชื่อมต่อกับโมเดลและการใช้งานจริง (Real-Time Inference)

ไฟล์: `test_optimized.py`

- โหลดโมเดลที่ฝึกแล้ว (`.pth`)
- ใช้ OpenCV Haar Cascade ตรวจจับใบหน้าผ่านกล้องเว็บแคม
- ตัดภาพใบหน้า → ปรับขนาดเป็น 48×48 → แปลงเป็น Tensor
- ส่งภาพเข้าสู่โมเดลเพื่อทำนายอารมณ์
- แสดงผลลัพธ์เป็นชื่ออารมณ์บนภาพแบบเรียลไทม์

## 5. อธิบาย Dataset ที่ใช้ และวิธีในการ Train

### Dataset ที่ใช้

- **ชื่อชุดข้อมูล:** FER2013 (Facial Expression Recognition 2013)
  **แหล่งที่มา:** Kaggle
  https://www.kaggle.com/code/gauravsharma99/facial-emotion-recognition
- **รูปแบบข้อมูล:**
  - บันทึกเป็น `.csv` โดยมีคอลัมน์หลักคือ
    - `emotion` → label ของอารมณ์ (0–6)
    - `pixels` → ค่าพิกเซลของภาพใบหน้าขนาด 48×48
    - `Usage` → ประเภทของข้อมูล (Training, PublicTest, PrivateTest)

| Emotion Label | Emotion Name |
|---|---|
| 0 | Angry |
| 1 | Disgust |
| 2 | Fear |
| 3 | Happy |
| 4 | Sad |
| 5 | Surprise |

| | |
|---|---|
| 6 | Neutral |

- **จำนวนภาพทั้งหมด:** 35,887 ภาพ
- **รูปแบบภาพ:** Grayscale (1 channel), ขนาด 48×48 pixels
- **การแบ่งข้อมูล:**
    - Training set: 28,709 ภาพ
    - Validation set (PublicTest): 3,589 ภาพ
    - Test set (PrivateTest): 3,589 ภาพ

## อธิบายโค้ดส่วนที่ใช้ Train

```
import torch.optim as optim
import torch.nn as nn
# --- (สมมติว่า model, train_loader, test_loader, device อยู่ตรงนี้แล้ว) ---
model = SimpleCNN_with_BatchNorm().to(device) # (จากโค้ดก่อนหน้า)
train_loader # (จากโค้ดก่อนหน้า)
test_loader # (จากโค้ดก่อนหน้า)
device # (จากโค้ดก่อนหน้า)
# ----------------------------------------------------------------------

# กำหนด Loss and Optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# --- ลูปการเทรน ---
num_epochs = 50

# (เผื่อเก็บ history ไว้พล็อตกราฟ)
train_losses = []
val_losses = []
```

```
val_accuracies = []

for epoch in range(num_epochs):

  # ==========================
  #     TRAINING LOOP
  # ==========================
  model.train() # ตั้งค่าเป็น training mode
  running_loss = 0.0

  for images, labels in train_loader:
    images, labels = images.to(device), labels.to(device)
    optimizer.zero_grad()
    outputs = model(images)
    loss = criterion(outputs, labels)
    loss.backward()
    optimizer.step()
    running_loss += loss.item() * images.size(0)

  epoch_loss = running_loss / len(train_dataset)
  train_losses.append(epoch_loss)

  # ==========================
  #     VALIDATION LOOP  <--- (ส่วนที่เพิ่มเข้ามา)
  # ==========================
  model.eval() # <--- สำคัญมาก! ตั้งค่าเป็น evaluation mode (ปิด Dropout, BatchNorm)
  val_running_loss = 0.0
  correct_preds = 0
  total_preds = 0

  with torch.no_grad(): # <--- สำคัญมาก! ไม่ต้องคำนวณ gradient ตอนเทส
    for images, labels in test_loader:
      images, labels = images.to(device), labels.to(device)

      # Forward pass
```

```
        outputs = model(images)

        # Calculate loss
        loss = criterion(outputs, labels)
        val_running_loss += loss.item() * images.size(0)

        # Calculate accuracy
        # torch.max จะคืนค่า (max_value, max_index)
        _, predicted_labels = torch.max(outputs, 1)

        total_preds += labels.size(0)
        correct_preds += (predicted_labels == labels).sum().item()

    # คำนวณ Loss และ Accuracy ของ Validation set
    epoch_val_loss = val_running_loss / len(test_dataset)
    epoch_val_acc = (correct_preds / total_preds) * 100

    val_losses.append(epoch_val_loss)
    val_accuracies.append(epoch_val_acc)

    # พิมพ์ผลลัพธ์ของ Epoch นี้
    print(f"Epoch [{epoch+1}/{num_epochs}], "
        f"Train Loss: {epoch_loss:.4f}, "
        f"Val Loss: {epoch_val_loss:.4f}, "
        f"Val Acc: {epoch_val_acc:.2f}%")


print("Training Finished!")

# บันทึกโมเดล
torch.save(model.state_dict(), 'fer2013_cnn_model_with_evaluate.pth') # (ผมแก้ชื่อไฟล์เล็กน้อย)
print("Model Saved!")
```

1. การวางโครงสร้าง Training (Training Loop)

- ลูปการเทรนทั้งหมดมี 50 epochs
- แต่ละ epoch แบ่งการทำงานเป็น 2 ขั้นตอนหลัก
    - Training Phase → ให้โมเดลเรียนรู้จากข้อมูลฝึก
    - Validation Phase → ทดสอบความแม่นยำกับข้อมูลทดสอบ

2. Training Phase

ตั้งค่าโมเดลให้อยู่ในโหมด training ด้วยคำสั่ง model.train()

- เพื่อเปิดการทำงานของ BatchNorm และ Dropout
- วนลูปผ่านข้อมูลใน train_loader (batch size = 64)
- ในแต่ละ batch มีลำดับการทำงานดังนี้
    - Forward Pass:
        - ส่งข้อมูลภาพ (images) เข้าโมเดล → ได้ผลลัพธ์ outputs
    - Compute Loss:
        - คำนวณค่าความผิดพลาดระหว่าง outputs กับ labels

        Backward Pass:
        - เคลียร์ค่า gradient เดิม
        - คำนวณ gradient ใหม่จาก loss
        - อัปเดตค่าน้ำหนักของโมเดล
- หลังจากประมวลผลครบทุก batch จะคำนวณ Train Loss เฉลี่ยของ epoch นั้น และบันทึกลงในตัวแปร train_losses

3. Validation Phase

- ตั้งค่าโมเดลเป็นโหมด evaluation เพื่อปิด Dropout และทำให้ BatchNorm ใช้ค่าเฉลี่ยคงที่
- ใช้ torch.no_grad() เพื่อปิดการคำนวณ gradient (ช่วยลดหน่วยความจำ)
- วนลูปผ่านข้อมูลใน test_loader
    - ทำการ Forward Pass เพื่อคำนวณผลลัพธ์
    - คำนวณค่า Validation Loss ด้วย criterion(outputs, labels)
    - นับจำนวนคำทำนายที่ถูกต้องเพื่อนำไปคำนวณ Accuracy (%)
- หลังจบแต่ละ epoch จะบันทึก
    - val_losses → ค่า Loss ของ validation
    - val_accuracies → ค่าความแม่นยำ (%) ของ validation

4. การแสดงผลระหว่างการเทรน

- หลังจบแต่ละ epoch จะพิมพ์ผลลัพธ์สรุปออกมาบนหน้าจอ เช่น
  Epoch [1/50], Train Loss: 1.8234, Val Loss: 1.5910, Val Acc: 44.52%
- เพื่อให้เห็นแนวโน้มว่าโมเดลเรียนรู้ดีขึ้นหรือไม่

5. การบันทึกโมเดล (Model Saving)

- เมื่อเทรนครบทุก epoch จะบันทึกน้ำหนักโมเดลลงในไฟล์
  torch.save(model.state_dict(),'fer2013_cnn_model_with_evaluate.pth')
- ไฟล์นี้จะถูกนำไปใช้ในขั้นตอนทำนายจริง (Real-Time Emotion Detection)

6. สรุปแนวทางการทำงานของโค้ด

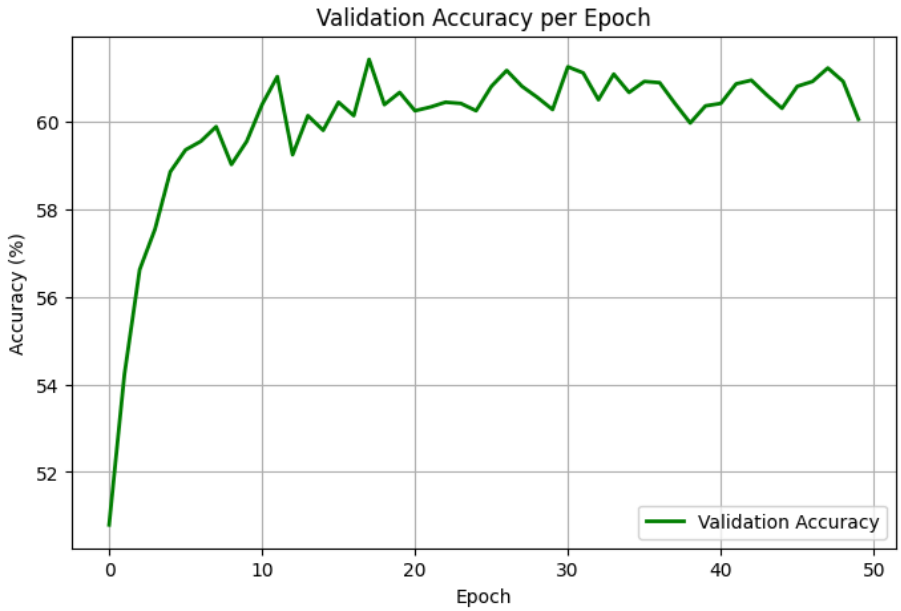
- Forward Pass → โมเดลทำนายผลลัพธ์จากภาพ
- Backward Pass → ปรับค่าน้ำหนักตามค่า Loss
- Evaluation → ตรวจสอบประสิทธิภาพบนชุดทดสอบ
- Saving → เก็บโมเดลที่เรียนรู้เสร็จแล้วไว้ใช้งานจริง

## 6. อธิบายการประเมิน (Evaluate Model)

### 1. การประเมินค่า Loss จากการเทรน

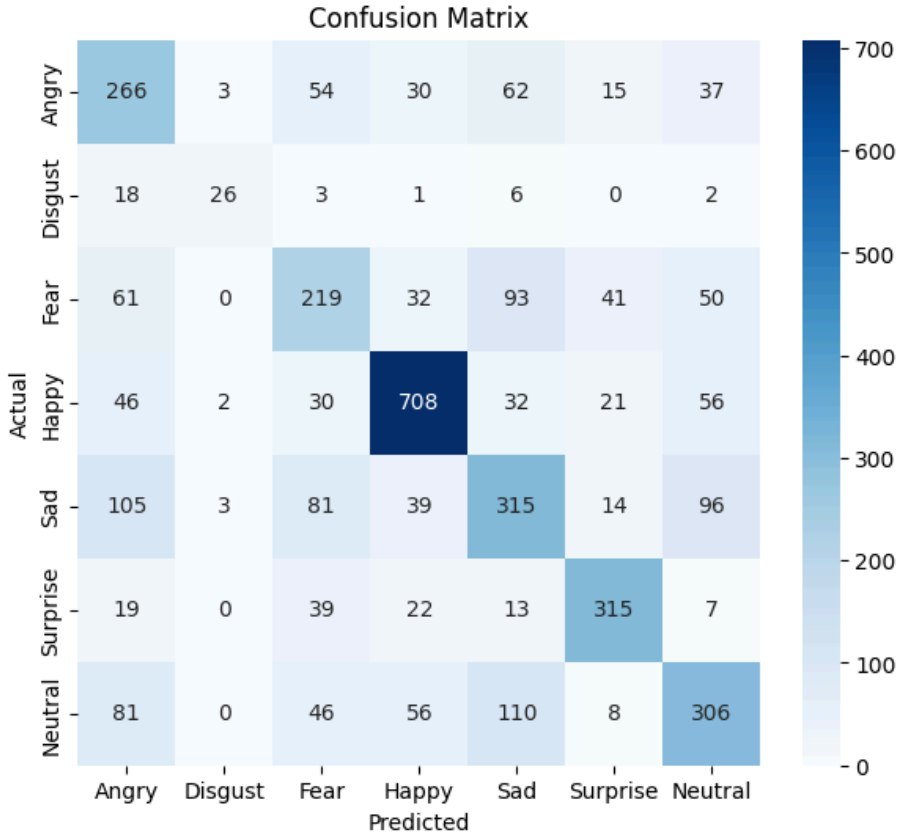

- ระหว่างการฝึก ได้เก็บค่าการเปลี่ยนแปลงของ Training Loss และ Validation Loss ไว้ในแต่ละ Epoch
- จากกราฟจะเห็นว่า
    - ค่า Training Loss ลดลงอย่างต่อเนื่อง แสดงว่าโมเดลเรียนรู้จากข้อมูลฝึกได้ดี
    - ค่า Validation Loss มีแนวโน้มเพิ่มขึ้นหลัง Epoch ที่ ~20 บ่งบอกถึงการเริ่มเกิด Overfitting
- ค่า Loss ใช้ในการตรวจสอบว่าการเรียนรู้ของโมเดลมีเสถียรภาพหรือไม่ และช่วยกำหนดจุดที่เหมาะสมในการหยุดฝึก (Early Stopping)

## 2. การประเมินค่า Accuracy




- ใช้คำนวณจากสัดส่วนของจำนวนภาพที่โมเดลทำนายถูกต้องต่อจำนวนภาพทั้งหมดในชุดทดสอบ
- ในการเทรนครั้งนี้ Accuracy ถูกคำนวณทุก Epoch แล้วเก็บไว้ในตัวแปร val_accuracies
- จากกราฟแสดงให้เห็นว่า Accuracy เพิ่มขึ้นรวดเร็วในช่วงต้น และคงที่อยู่ราว 60–62% หลังจาก Epoch ที่ 10 เป็นต้นไป
- แสดงว่าโมเดลสามารถแยกแยะอารมณ์ได้ในระดับที่มีความถูกต้องพอสมควรสำหรับงานจำแนกภาพอารมณ์

### 3. การประเมินด้วย Confusion Matrix




- ใช้เพื่อแสดงการจำแนกผลลัพธ์จริงกับผลลัพธ์ที่โมเดลทำนาย
- ในแต่ละแถวคือ “อารมณ์จริง” และแต่ละคอลัมน์คือ “อารมณ์ที่โมเดลทำนายได้”
- จาก Matrix พบว่าโมเดลสามารถจำแนกอารมณ์ Happy, Neutral และ Surprise ได้ค่อนข้างดี (ค่าบนแนวทแยงสูง) แต่มีการสับสนระหว่าง Fear, Sad และ Angry ซึ่งมักมีลักษณะทางสีหน้าใกล้เคียงกัน

## 7. อธิบายบทความอ้างอิงและงานที่เกี่ยวข้อง

โครงงานนี้ได้แนวคิดและแนวทางการพัฒนาโมเดลมาจากบทความ “Facial Emotion Recognition” โดย Gaurav Sharma เผยแพร่บนเว็บไซต์ Kaggle (https://www.kaggle.com/code/gauravsharma99/facial-emotion-recognition)

**1. ข้อมูลจากบทความต้นฉบับ**

- Dataset: FER2013 (Facial Expression Recognition 2013)
  มีภาพขาวดำขนาด 48×48 พิกเซล จำนวน 35,887 ภาพ แบ่งเป็น 7 อารมณ์ ได้แก่ Angry, Disgust, Fear, Happy, Sad, Surprise และ Neutral
- สถาปัตยกรรมโมเดล:
  ใช้ CNN แบบหลายชั้น (Conv2D → ReLU → MaxPooling → Dropout)
  พร้อม Softmax Output Layer สำหรับจำแนกอารมณ์
- เทคนิคการฝึกโมเดล:
  ใช้ Adam Optimizer กับ Categorical CrossEntropy Loss
  และเพิ่ม Data Augmentation เช่น การพลิกภาพ (flip) และหมุน (rotation)
  เพื่อเพิ่มความหลากหลายและลด Overfitting

**2. แนวคิดที่นำมาประยุกต์ใช้ในโครงงานนี้**

- นำแนวคิดสถาปัตยกรรม CNN หลักจากบทความต้นฉบับมาพัฒนาใหม่ให้เหมาะกับ PyTorch Framework
- เพิ่ม Batch Normalization ในแต่ละ Convolution Block เพื่อให้การเรียนรู้เสถียรขึ้น
- เพิ่มขั้นตอน Validation Phase และการเก็บค่า Loss / Accuracy ราย Epoch เพื่อใช้วิเคราะห์ประสิทธิภาพของโมเดล
- ใช้หลักการเดียวกันในการตรวจจับใบหน้าจริงผ่านกล้อง โดยใช้ไฟล์ haarcascade_frontalface_default.xml และโมเดลที่ฝึกจาก FER2013

### 3. ความเกี่ยวข้องกับงานวิจัยอื่น

- Kaggle Notebook ดังกล่าวต่อยอดแนวคิดจากงานของ Goodfellow et al. (2013) และ Tang (2013) ที่เปิดตัว FER2013 dataset และใช้ CNN เป็นสถาปัตยกรรมหลัก

## 8.ลิงค์ Github ไว้รวบรวมโค้ด

https://github.com/Napatsanan-dmp/Project_deeplearning

## 9.แบ่งสัดส่วนการทำงาน

1.นางสาวนภัสนันท์ ดามะพร (50%)

- ศึกษาและรวบรวมข้อมูลเกี่ยวกับ Facial Emotion Recognition และ FER2013 Dataset
- ออกแบบและสร้างสถาปัตยกรรมโมเดล CNN ใน PyTorch
- พัฒนา Notebook สำหรับการเทรนและประเมินผลโมเดล
- เขียนรายงานส่วนการอธิบายโมเดล, วิธีการเทรน

2.นายสรัล สังข์วร (50%)

- พัฒนาโค้ดส่วนการตรวจจับใบหน้าและเชื่อมต่อโมเดลกับกล้อง (test_optimized.py)
- ปรับแต่งการทำงานของโมเดลสำหรับการใช้งานแบบ Real-Time
- วิเคราะห์ผลการทำงานของโมเดลจากค่า Accuracy, Precision, Recall, และ F1-score
- เขียนรายงานส่วนการอธิบายการใช้งานจริง, การเชื่อมต่อระบบ, และการประเมินผล