

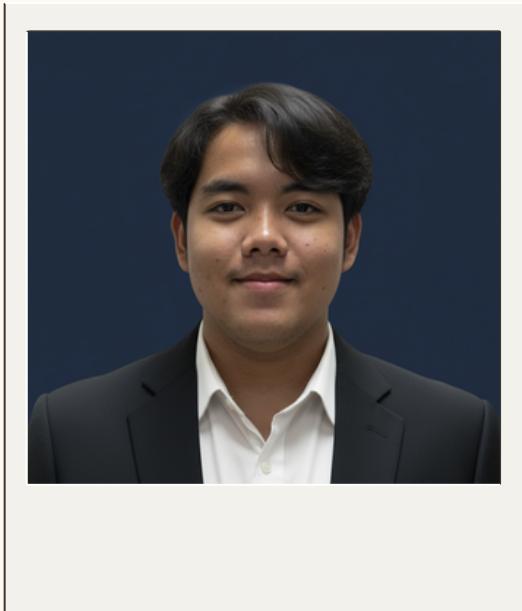
Face Recognition 2013



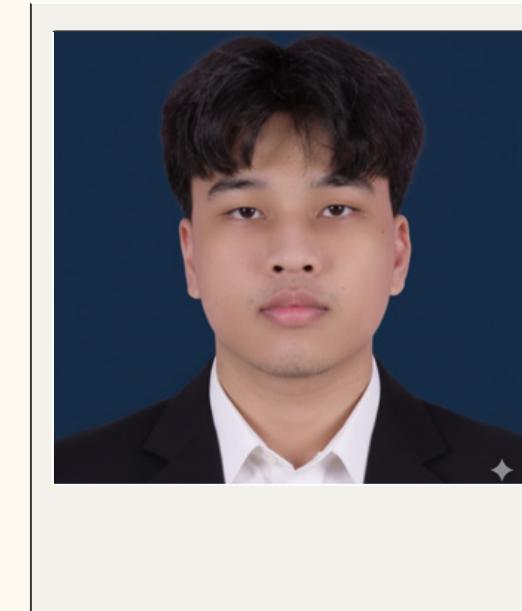
Using Five Layers Model Architecture



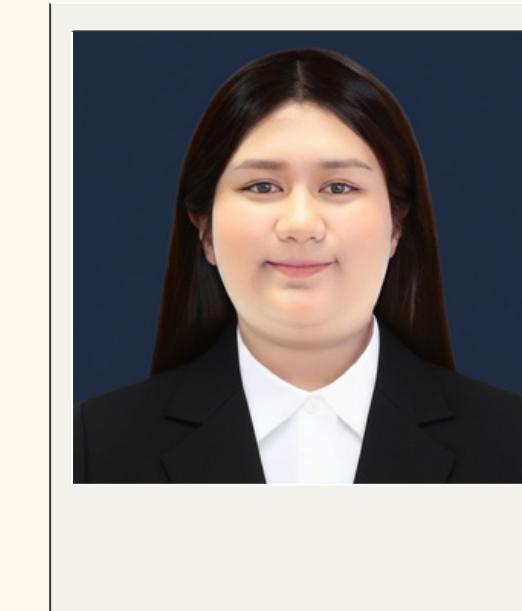
Team member



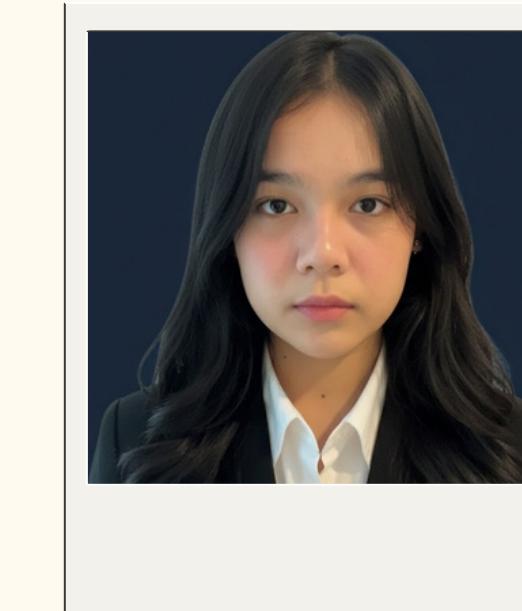
จิรา�ุ ชูนตร
6630251059



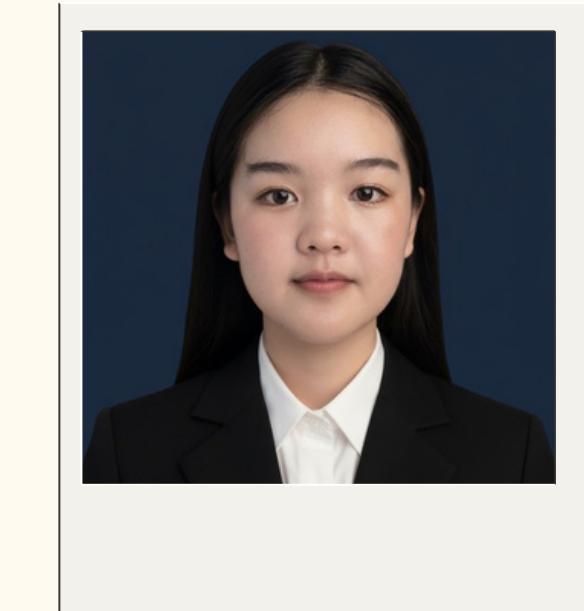
วิชยุตม์ วงศ์ชัย
6630251229



Kasitorn Sangpech
6630251237



นลินรรณ์ โสภานิริยารณ์
6630251334



สาวลักษณ์ ประภาศิริสุล
6630251423

+Dataset

Dataset: FER2013 Facial Expression Dataset



- Dataset สำหรับงานจำแนกอารมณ์จากใบหน้า
- ภาพเป็น ขาวดำ (Grayscale) ขนาด 48×48 pixels
- แบ่งออกเป็น 7 อารมณ์
(เช่น Angry, Happy, Sad, Fear, Surprise, Neutral, Disgust)
- แบ่งเป็น Train / Validation / Test

Project Goal

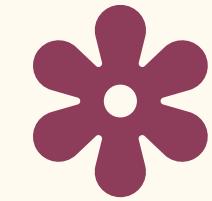
- **test accuracy > 65%**
- **test validation > 65%**





CNN Model Design and Architecture

- ใช้ Five Layers Model Architecture ตาม Research Paper
- ปรับเปลี่ยน Hyperparameter Tuning
- ปรับเปลี่ยน Data Augmentation



Training Strategy & Learning Control



- ใช้ CrossEntropyLoss + Label Smoothing เพื่อลดความมั่นใจเกินไปของโมเดล
- ใช้ Class Weight เพื่อแก้ปัญหา class imbalance
- ใช้ AdamW Optimizer พร้อม weight decay เพื่อเพิ่ม generalization
- ใช้ ReduceLROnPlateau ลด learning rate เมื่อ validation loss ไม่ดีขึ้น
- ใช้ Early Stopping เพื่อป้องกัน overfitting
- ฝึกโมเดลสูงสุด 100 epochs และบันทึกโมเดลที่ดีที่สุดจาก validation loss



Data Augmentation

- **Random Horizontal Flip** กลับภาพแนวอนแบบสุ่ม
- **Random Rotation** หมุนภาพแบบสุ่ม
- **Random Affine** หมุน $+-10\%$ เลื่อนภาพได้ 10% ของภาพ, ขยาย
- **Random Erasing** ช่วยลด overfitting และเพิ่มความทนทานของโมเดล

Model Architecture

```
class CNNModel(nn.Module):
    def __init__(self, num_classes=7):
        super(CNNModel, self).__init__()

        self.conv1 = nn.Conv2d(1, 128, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(128)
        self.pool1 = nn.MaxPool2d(2, 2)
        self.dropout1 = nn.Dropout(0.25)

        self.conv2 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(256)
        self.pool2 = nn.MaxPool2d(2, 2)
        self.dropout2 = nn.Dropout(0.25)

        self.conv3 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(512)
        self.pool3 = nn.MaxPool2d(2, 2)
        self.dropout3 = nn.Dropout(0.25)

        self.conv4 = nn.Conv2d(512, 1024, kernel_size=3, padding=1)
        self.bn4 = nn.BatchNorm2d(1024)
        self.pool4 = nn.MaxPool2d(2, 2)
        self.dropout4 = nn.Dropout(0.25)

        self.conv5 = nn.Conv2d(1024, 1024, kernel_size=3, padding=1)
        self.bn5 = nn.BatchNorm2d(1024)
        self.dropout5 = nn.Dropout(0.25)

        self.global_avg_pool = nn.AdaptiveAvgPool2d(1)

        self.fc1 = nn.Linear(1024, 512)
        self.bn_fc1 = nn.BatchNorm1d(512)
        self.dropout_fc1 = nn.Dropout(0.5)

        self.fc2 = nn.Linear(512, 256)
        self.bn_fc2 = nn.BatchNorm1d(256)
        self.dropout_fc2 = nn.Dropout(0.5)

        self.fc3 = nn.Linear(256, num_classes)
```

Five Layers Model ดังภาพเป็นอันเดียวกับที่อยู่ใน Research
ประกอบด้วย

Conv2d → 128-1024 ในแต่ละชั้น

BatchNorm → 128 - 1024 ในแต่ละชั้น

Pool → 2,2 ในแต่ละชั้น 1-5

Dropout → 0.25 ในแต่ละ Conv2d , และ 0.5 ใน FCL

ใน FCL layer 3 ชั้น ประกอบด้วย

fc1 → Linear() , BatchNorm() , Dropout()

fc2 → Linear() , BatchNorm() , Dropout()

fc3 → Linear,num_classes



Data Augmentation

```
train_transform = v2.Compose([
    v2.ToImage(),
    v2.RandomHorizontalFlip(p=0.5),
    v2.RandomRotation(degrees=10),
    v2.RandomAffine(
        degrees=10,
        translate=(0.1, 0.1),
        scale=(0.9, 1.1)
    ),
    v2.RandomErasing(p=0.3, scale=(0.02, 0.25)),
    v2.ToDtype(torch.float32, scale=True),
    v2.Normalize(mean=[0.5], std=[0.5]),
])

val_test_transform = v2.Compose([
    v2.ToImage(),
    v2.ToDtype(torch.float32, scale=True),
    v2.Normalize(mean=[0.5], std=[0.5]),
])
```

- ToImage แปลงภาพ pixel → tensor
- Random Horizontal Flip กลับภาพแนว
แนวนอนแบบสุ่ม
- Random Rotation หมุนภาพแบบสุ่ม
- Random Affine หมุน +-10% เลื่อนภาพ
ได้ 10% ของภาพ, ขยาย
- Random Erasing ช่วยลด overfitting
และเพิ่มความกันทานของโมเดล
- ToDtype → uint8 > float32 , scale
 $0,255 \rightarrow 0,1$



Hyperparameter Tuning

```
class_weights = compute_class_weight('balanced', classes=np.unique(y_train_enc), y=y_train_enc)
class_weights = torch.tensor(class_weights, dtype=torch.float32).cuda()
```

```
criterion = nn.CrossEntropyLoss(label_smoothing=0.1, weight=class_weights)
optimizer = optim.AdamW(model.parameters(), lr=0.0008, weight_decay=1e-4)
scheduler = ReduceLROnPlateau(
```

```
    optimizer,
    mode='min',
    factor=0.6,
    patience=3,
    min_lr=1e-5
)
```

```
num_epochs = 100
best_val_loss = float("inf")
```

```
patience = 10
counter = 0
```



Class Weight Compute Balanced หมายถึงกำให้ทุกคลาส ใช้ class weight โดยมี metric เป็น balanced ก็คือ weight ทุกคลาสจะเท่ากันเสมอ

nn.Crossentropy ใช้กับ classification หลายคลาส
label_smoothing = 0.1 ลด overconfident, generalize

weight=class_weights = เรียกใช้ class_weights

AdamW = Adam + Weight_decay

LR=0.0008 = Learning rate ระดับปานกลาง

weight_decay = 0.0001 เพื่อลด overfitting

scheduler = สังเกตว่า val_loss ถ้าไม่ลดเกิน 3 epoch ให้ลด

LR= 0.6 x LR เดิม แต่ต่ำสุดไม่เกิด 0.00001

num_epochs = 100 จำนวนรอบที่เทรน
patience = 10 ถ้า val_loss ไม่ดีขึ้น 10 รอบติดต่อกันให้เลิกเทรน
counter = 0 คือตัวใช้บันทึกว่า val_loss ไม่ดีขึ้น 1 รอบให้ counter+1



Final Results

Metric	Value
Accuracy	0.6718
Precision	0.6738
Recall	0.6718
F1-score	0.6685
Inference Time	1.69 seconds



8. Limitation and Improvement

ข้อจำกัดของโมเดล

- เกิด Overfitting บางส่วน
- Class Imbalance โดยเฉพาะคลาส Disgust

แนวทางพัฒนาในอนาคต

- ทำการปรับ Hyperparameter Tuning ที่ดีกว่า
- เพิ่มข้อมูล (Data Balancing)
- ใช้ Pretrained Model เช่น ResNet, MobileNet
- ปรับ Data Augmentation ให้หลากหลายขึ้น



Thank you !

Question & answer