EGCO486: Image Processing
Final Project

RE-AL or FA-KE

Reported by

Nipatsa Chainiwattana 6513114
Puthipong Yomabut 6513134
Phattaradanai Sornsawang 6513172
Patiharn Kamenkit 6513170

Submitted to

Asst. Prof. Dr. Narit Hnoohom

COMPUTER ENGINEERING INTERNATIONAL PROGRAM
MAHIDOL UNIVERSITY INTERNATIONAL COLLEGE

2024

Background

จากปัจจุบันเทคโนโลยี Generative AI มีบทบาทสำคัญในหลายด้านโดยเฉพาะในการสร้างภาพที่มีความเหมือนจริงสูง ทำให้ยากต่อการแยกแยะระหว่างภาพวาดที่ถูกสร้างขึ้นด้วย Generative AI และภาพที่เป็น ของจริง คณะผู้จัดทำจึงได้จัดทำโปรเจคนี้ขึ้นเพื่อแก้ปัญหาการแอบอ้างใช้ Generative AI แทนภาพจริง โดย RE-AL or FA-KE เป็นเว็บแอพพลิเคชั่นสำหรับจำแนกภาพวาดจริงกับภาพวาดจาก Generative AI โดยใช้หลักการทาง Image Processing และ Convolutional Neural Network มาประยุกต์ใช้ โดยในโปรเจคนี้ได้พัฒนาโมเดล Convolutional Neural Networks (CNN) และโมเดลที่ใช้เทคนิค Transfer Learning จากโมเดล ResNet50, Moblienet V2 และ VGG19 แล้ว Deploy บนเว็บแอพพลิเคชั่นที่พัฒนาด้วย Streamlit เพื่อใช้แสดงผลลัพธ์และทำนายรูปภาพที่อัพโหลดว่าเป็นภาพวาดจริงหรือภาพวาดจาก Generative AI คณะผู้จัดทำขอขอบพระคุณ ผศ.ดร. นริศ หนูหอม ผู้ให้ความรู้ และแนวทางในการศึกษา สุดท้ายนี้ทางคณะผู้จัดทำหวังว่ารายงานฉบับนี้จะให้ความรู้และประโยชน์ไม่มากก็น้อยแก่ผู้อ่านทุกท่าน โดยหากมีข้อผิดพลาดประการใดผู้จัดทำจะขอน้อมรับไว้และขออภัยมา ณ ที่นี้

คณะผู้จัดทำ

Contents

Objective

1. เพื่อสามารถจำแนกภาพวาดที่สร้างโดยมนุษย์ (Real Art) และภาพวาดที่สร้างโดย AI (AI-Generated) โดยใช้โมเดล Deep Learning
2. แสดงผลบน Web Application ผ่าน Streamlit platform ซึ่งเป็นเครื่องมือที่ใช้การสร้างและแชร์ Web Application เพื่อให้ user สามารถใช้งานได้ง่าย
3. เพื่อนำความรู้ด้าน Image Processing และ Deep Learning Processing มาประยุกต์ใช้ในการแก้ปัญหาของวงการศิลปะ โดยเฉพาะในกรณีที่เกี่ยวกับการแอบอ้างหรือปลอมแปลงผลงานและในเรื่องของข้อจำกัดทางลิขสิทธิ์จากภาพที่สร้างจาก AI

Scope

โปรเจคนี้เน้นศึกษาในการพัฒนาเว็บแอพพลิเคชั่นและโมเดลที่ใช้ในการจำแนกภาพระหว่างภาพวาดจริงและภาพวาดจาก Generative AI Dataset โดยพัฒนาโมเดลด้วย Dataset จาก Kaggle : AI and Human Art Classification โดยได้วางแผนพัฒนาโมเดล 5 แบบ ดังนี้ :

1. Basic CNN Classifier
2. ResNet50 Classifier
3. ResNet50 Fine-Tuned Classifier
4. MobileNetV2 Classifier
5. VGG19 Classifier

Method

ขั้นตอนการพัฒนาโมเดลที่ 1 (Basic CNN Classifier)

1. Import Library ที่ใช้งาน

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D , Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.utils.class_weight import compute_class_weight
import itertools
import os
import matplotlib.pyplot as plt
import cv2
import random
import shutil
import requests
from sklearn.model_selection import train_test_split
%matplotlib inline
```

2. กำหนด path และสร้างฟังก์ชั่นสำหรับแบ่งชุดข้อมูลเป็น train, validation และ test

```
real_path = 'data/real'
fake_path = 'data/fake'
```

```
base_dir = '/data'
train_dir = os.path.join(base_dir, 'train')
val_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

for directory in [train_dir, val_dir, test_dir]:
    os.makedirs(os.path.join(directory, 'real'), exist_ok=True)
    os.makedirs(os.path.join(directory, 'fake'), exist_ok=True)
```

```
split_data(real_path, os.path.join(train_dir, 'real'), os.path.join(val_dir, 'real'), os.path.join(test_dir, 'real'))
split_data(fake_path, os.path.join(train_dir, 'fake'), os.path.join(val_dir, 'fake'), os.path.join(test_dir, 'fake'))
```

3. ใช้ ImageDataGenerator ในการเตรียมข้อมูลภาพด้วยฟังก์ชั่น preprocess_input จาก ResNet50 โดยกำหนดขนาดภาพเป็น 224 x 224 และทำ Data Augmentation เพื่อให้โมเดลได้เรียนรู้ภาพในลักษณะหลายๆแบบ โดยแบ่งข้อมูลเป็น train_batches, valid_batches, test_batches โดยกำหนดขนาดแบตช์ (batch size) ละ 16 ภาพ

```
batch_size = 16
img_size = (224, 224)

data_gen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    horizontal_flip=True,
    vertical_flip=True,
    brightness_range=[0.4,1.5],
    zoom_range=0.3,
    rotation_range=40,
    fill_mode='nearest'
)

train_batches = data_gen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
)

valid_batches = data_gen.flow_from_directory(
    val_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
)

test_batches = data_gen.flow_from_directory(
    test_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 26139 images belonging to 2 classes.
Found 5943 images belonging to 2 classes.
Found 5934 images belonging to 2 classes.
```

4. คำนวน weight ของแต่ละ class เพื่อแก้ปัญหา class imbalance จากการที่จำนวนรูปของทั้ง 2 class ไม่เท่ากัน

```
class_labels = np.unique(train_batches.classes)
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=class_labels,
    y=train_batches.classes
)
class_weights_dict = dict(enumerate(class_weights))
print("Class Weights:", class_weights_dict)
```

```
Class Weights: {0: 0.7144926743931773, 1: 1.6655409710717473}
```

5. สร้างโมเดล CNN ที่มี layer ดังนี้ โดยเลือกใช้ output layer ที่มี activation เป็น softmax เพื่อแสดงค่าความน่าจะเป็นของทั้ง 2 คลาส

```
model = Sequential([
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same', input_shape=(224,224,3)),
    MaxPool2D(),
    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(),
    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    Conv2D(filters=256, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(),
    layers.GlobalAveragePooling2D(),
    Dense(units=128, activation='relu'),
    Dropout(0.2),
    Dense(units=64, activation='relu'),
    Dropout(0.2),
    Dense(units=2, activation='softmax')
])
```

6. ใช้คำสั่ง model.summary() เพื่อตรวจสอบจำนวนเลเยอร์และพารามิเตอร์ ได้ผลดังนี้

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 112, 112, 64)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 128)     73856

 conv2d_2 (Conv2D)           (None, 112, 112, 128)     147584

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 128)      0
 2D)

 conv2d_3 (Conv2D)           (None, 56, 56, 256)       295168

 conv2d_4 (Conv2D)           (None, 56, 56, 256)       590080

 max_pooling2d_2 (MaxPooling  (None, 28, 28, 256)      0
 2D)

 global_average_pooling2d (G  (None, 256)              0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               32896

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 2)                 130

=================================================================
Total params: 1,149,762
Trainable params: 1,149,762
Non-trainable params: 0
```

7. Compile โมเดลโดยใช้ Optimizer เป็น adam และ Loss function เป็น categorical_crossentropy และเพิ่ม metrics accuracy สำหรับการวัดผล

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

8. เทรนโมเดลด้วยคำสั่ง model.fit โดยทดลองทั้งแบบ 5 epoch และ 10 epoch เพื่อเปรียบเทียบผลลัพธ์ที่ได้

```
history = model.fit(x=train_batches,
    validation_data=valid_batches,
    epochs=5,
    class_weight=class_weights_dict,
    verbose=1,
)
```

```
Epoch 1/5
  59/1634 [>.............................] - ETA: 7:55 - loss: 1.0328 - accuracy: 0.5138
C:\Users\Phattaradanai\anaconda3\envs\tenserflowgpu\lib\site-packages\PIL\Image.py:1056: UserWarning: Palette images with Transparency expressed in bytes
should be converted to RGBA images
  warnings.warn(
1634/1634 [==============================] - 805s 471ms/step - loss: 0.6942 - accuracy: 0.5529 - val_loss: 0.6563 - val_accuracy: 0.6450
Epoch 2/5
1634/1634 [==============================] - 747s 457ms/step - loss: 0.5923 - accuracy: 0.7033 - val_loss: 0.5303 - val_accuracy: 0.7892
Epoch 3/5
1634/1634 [==============================] - 685s 419ms/step - loss: 0.5153 - accuracy: 0.7767 - val_loss: 0.4890 - val_accuracy: 0.7945
Epoch 4/5
1634/1634 [==============================] - 728s 446ms/step - loss: 0.4905 - accuracy: 0.7944 - val_loss: 0.5421 - val_accuracy: 0.7459
Epoch 5/5
1634/1634 [==============================] - 755s 462ms/step - loss: 0.4733 - accuracy: 0.8024 - val_loss: 0.5022 - val_accuracy: 0.7829
```

9. นำ model ไป predict กับข้อมูลชุด test แล้วตรวจสอบความแม่นยำที่ได้ด้วย classification_report

```
predictions = model.predict(x=test_batches, steps=len(test_batches), verbose=1)
```

```
116/371 [========>.....................] - ETA: 1:55
C:\Users\Phattaradanai\anaconda3\envs\tenserflowgpu\lib\site-packages\PIL\Image.py:1056: UserWarning: Palette images with Transparency expressed in bytes
should be converted to RGBA images
  warnings.warn(
371/371 [==============================] - 156s 419ms/step
```

```
print(classification_report(test_batches.classes,np.argmax(predictions, axis = -1)))
```

```
              precision    recall  f1-score   support

           0       0.93      0.75      0.83      4153
           1       0.59      0.86      0.70      1781

    accuracy                           0.78      5934
   macro avg       0.76      0.80      0.76      5934
weighted avg       0.83      0.78      0.79      5934
```

10. นำ model ไป predict กับข้อมูลภาพที่รวบรวมจากอินเตอร์เน็ตเพื่อตรวจสอบความแม่นยำของโมเดลเมื่อใช้กับภาพนอก Dataset แล้วประเมินด้วย classification_report

```
test2_dir = 'data/test2/'

test_batches2 = data_gen.flow_from_directory(
    test2_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 40 images belonging to 2 classes.
```

```
predictions = model.predict(x=test_batches2, steps=len(test_batches2), verbose=1)
```

```
C:\Users\Phattaradanai\anaconda3\envs\tenserflowgpu\lib\site-packages\PIL\Image.py:1056: UserWarning: Palette images with Transparency expressed in bytes
should be converted to RGBA images
  warnings.warn(
3/3 [==============================] - 14s 2s/step
```

```
print(classification_report(test_batches2.classes,np.argmax(predictions, axis = -1)))
```

```
              precision    recall  f1-score   support

           0       0.51      0.90      0.65        20
           1       0.60      0.15      0.24        20

    accuracy                           0.53        40
   macro avg       0.56      0.53      0.45        40
weighted avg       0.56      0.53      0.45        40
```

11. เซฟไฟล์โมเดล เป็นไฟล์ .h5 เพื่อนำไป deploy บนเว็บ

```
model.save('modelcnnh.h5')
```

ขั้นตอนการพัฒนาโมเดลที่ 2 (ResNet50 Classifier)

ใช้กระบวนการเดียวกับการพัฒนาโมเดลแรก แต่เปลี่ยนในส่วนขั้นตอนการสร้างโมเดล ด้วยการ import ResNet50 มาเป็น base_model แล้วทำการ Freeze layer ไว้ แล้วเพิ่ม GlobalAveragePooling2D layer กับ output layer

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

```
for layer in base_model.layers:
    layer.trainable = False
```

```
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(2, activation='softmax')
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 2)                 4098

=================================================================
Total params: 23,591,810
Trainable params: 4,098
Non-trainable params: 23,587,712
_________________________________________________________________
```

ขั้นตอนการพัฒนาโมเดลที่ 3 (ResNet50 Fine-Tuned Classifier)

ใช้กระบวนการเดียวกับการพัฒนาโมเดลที่ 2 แต่เพิ่ม layer Dense กับ Dropout ก่อน output layer

เพื่อช่วยให้โมเดลทำนายได้ดีขึ้นและลด overfit

```
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

```
for layer in base_model.layers:
    layer.trainable = False
```

```
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    Dense(units=128, activation='relu'),
    Dropout(0.2),
    Dense(units=64, activation='relu'),
    Dropout(0.2),
    layers.Dense(2, activation='softmax')
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               262272

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_2 (Dense)             (None, 2)                 130

=================================================================
Total params: 23,858,370
Trainable params: 270,658
Non-trainable params: 23,587,712
_________________________________________________________________
```

ขั้นตอนการพัฒนาโมเดลที่ 4 (MobilenetV2 Fine-Tuned Classifier)

ใช้กระบวนการเดียวกับการพัฒนาโมเดลที่ 3 แต่ใช้ base model เป็น MobileNetV2 และใช้ preprocess_input จาก mobilenet_v2

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
```

```
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

ขั้นตอนการพัฒนาโมเดลที่ 5 (VGG19 Fine-Tuned Classifier)

ใช้กระบวนการเดียวกับการพัฒนาโมเดลที่ 3 แต่ใช้ base model เป็น VGG19 และใช้ preprocess_input จาก vgg19

```
from tensorflow.keras.applications import VGG19
from tensorflow.keras.applications.vgg19 import preprocess_input
```

```
base_model = VGG19(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

ขั้นตอนการพัฒนาเว็บแอพพลิเคชั่น

app.py

เป็นส่วนที่รวมหน้า page ต่างๆของ Web เข้าไว้ด้วยกัน

```python
import streamlit as st
import pandas as pd
import numpy as np
import streamlit_option_menu
from streamlit_option_menu import option_menu
from page.Home import show_home
from page.About import show_about
from page.Upload import show_upload
from page.Visual import show_visual

with st.sidebar:
    selected = option_menu(
        menu_title = 'Main Menu',
        options = ['Home', 'Visual', 'Upload', 'About us'],
        icons = ['house', 'eye', 'upload', 'info-circle'],
        menu_icon = 'cast',
        default_index = 0
    )

def switch_case(option):
    if option == 'Home':
        return show_home()
    elif option == 'Visual':
        return show_visual()
    elif option == 'Upload':
        return show_upload()
    else : return show_about()

def main():
    switch_case(selected)

if __name__=='__main__':
    main()
```

Home.py

เป็นหน้า Web หลักที่อธิบายเกี่ยวกับ Project นี้

```python
import streamlit as st

def load_css():
    with open('style.css') as f:
        st.markdown(f'<style>{f.read()}</style>', unsafe_allow_html=True)


def show_home():
    load_css()
    st.markdown('<h1 class="center-header"> RE-AL or FA-KE </h1>', unsafe_allow_html=True)
    st.markdown('<h3 class="center-header"> Image Classification</h3>', unsafe_allow_html=True)

    st.image("src/Home.jpeg", width=650)

    st.markdown('''<h6 class ="about-container-home"> <br>         These days, artificial intelligence (AI)
                into our daily lives, offering enormous advantages in many different domains. It does, however, have certain disadvantages. This is
                , unsafe_allow_html=True)
```

Visual.py

เป็นหน้าที่สุ่มรูปภาพที่ใช้ train model มาเข้า Model และ โชว์ผลลัพธ์จากการ predict

```
def show_visual():
    load_css()

    fake_folder = "src/fake"
    real_folder = "src/real"

    st.markdown('<h1 class="center-header">AI Image Example</h1>', unsafe_allow_html=True)

    image_path = get_random_image(fake_folder)
    print_image_visual(image_path)
    st.write("")
    col1, col2, col3 = st.columns(3)
    if image_path:
        with col1:
            st.markdown('<h4>Model 1</h4>', unsafe_allow_html=True)
            predict(image_path, load_model('Model 1'))
        with col2:
            st.markdown('<h4>Model 2</h4>', unsafe_allow_html=True)
            predict(image_path, load_model('Model 2'))
        with col3:
            st.markdown('<h4>Model 3</h4>', unsafe_allow_html=True)
            predict(image_path, load_model('Model 3'))
    else:
        st.write("No images found in fake folder.")
```

```
    st.markdown('<h1 class="center-header">Real Image Example</h1>', unsafe_allow_html=True)

    image_path = get_random_image(real_folder)
    print_image_visual(image_path)
    st.write("")
    col4, col5, col6 = st.columns(3)
    if image_path:
        with col4:
            st.markdown('<h4>Model 1</h4>', unsafe_allow_html=True)
            predict(image_path, load_model('Model 1'))
        with col5:
            st.markdown('<h4>Model 2</h4>', unsafe_allow_html=True)
            predict(image_path, load_model('Model 2'))
        with col6:
            st.markdown('<h4>Model 3</h4>', unsafe_allow_html=True)
            predict(image_path, load_model('Model 3'))
    else:
        st.write("No images found in fake folder.")
```

load_model() : เป็น function ที่เรียกใช้ Model ต่างๆ

```
@st.cache_resource
def load_model(option_model):
    if option_model is None:
            st.error("Please Select Model.")
            return
    if option_model == 'Model 1':
        model = tf.keras.models.load_model("Model1.h5")
    elif option_model == 'Model 2':
        model = tf.keras.models.load_model("Model2.h5")
    elif option_model == 'Model 3':
        model = tf.keras.models.load_model("Model3.h5")
    return model
```

visualize_graph() : **เป็น** function **ที่แสดงกราฟต่างๆ เช่น จำนวนรูปที่ใช้** train model **และ ค่า** Accuracy **และ** Loss **ในการ** train model **แต่ละรอบ**

```
def visualize_graph():
    labels = ['AI painting pictures', 'REAL painting pictures']
    sizes = [10330, 8288]
    colors = ['#E44C51', '#4C7AE4']


    fig = go.Figure(data=[go.Pie(
        labels=labels,
        values=sizes,
        textinfo='label+percent',
        insidetextorientation='radial',
        marker=dict(colors=colors),
    )])


    fig.update_layout(
        title={
            'text': "Segment of AI and Real Image",
            'y': 0.9,
            'x': 0.4,
            'xanchor': 'center',
            'yanchor': 'top'
        },
        title_font_size=24
    )
```

```
    ## Resnet
    epoch = [1, 2, 3, 4, 5]
    loss = [0.3782561421, 0.3311806023, 0.3202846348, 0.3099855781, 0.3165609241]
    accuracy = [0.8469719291, 0.8708825707, 0.8750143647, 0.8810971975, 0.8795669079]
    val_loss = [0.4360812604, 0.3923163712, 0.3274979591, 0.4336121678, 0.2789599895]
    val_accuracy = [0.8189466596, 0.833754003, 0.8623591065, 0.8244994283, 0.8921419978]



    accuracy_fig = go.Figure()
    accuracy_fig.add_trace(go.Scatter(x=epoch, y=accuracy, name='Training Accuracy',
                                      line=dict(color='#E49D4C', width=3, dash='dash')))
    accuracy_fig.add_trace(go.Scatter(x=epoch, y=val_accuracy, name='Validation Accuracy',
                                      line=dict(color='#E49D4C', width=3)))
    accuracy_fig.update_layout(
        title=dict(text='Accuracy of Resnet Model'),
        xaxis=dict(title=dict(text='Epoch')),
        yaxis=dict(title=dict(text='Accuracy')),
    )


    loss_fig = go.Figure()
    loss_fig.add_trace(go.Scatter(x=epoch, y=loss, name='Training Loss',
                                  line=dict(color='firebrick', width=3, dash='dash')))
    loss_fig.add_trace(go.Scatter(x=epoch, y=val_loss, name='Validation Loss',
                                  line=dict(color='firebrick', width=3)))
    loss_fig.update_layout(
        title=dict(text='Loss of Resnet Model'),
        xaxis=dict(title=dict(text='Epoch')),
        yaxis=dict(title=dict(text='Loss')),
    )
```

Upload.py เป็นหน้าที่ให้ **User** นำภาพมาเข้า **Model** ได้

```python
def show_upload():
    load_css()

    st.header('Detail of Model')
    st.markdown("""
    - Model 1 (Basic CNN Model) : Can classify images within the dataset but poor results when use with images outside the dataset.
    - Model 2 (ResNet50) : Can classify both images within the dataset and those outside the dataset.
    - Model 3 (ResNet50) with Fine-Tuning : Can classify both images within the dataset and those outside the dataset and usually better result that
    """)
    st.markdown('<div><h2>Convert image</h2></div>', unsafe_allow_html=True)
    st.markdown('<div><h5>You can reduce noise ,blur and increase light here </h5></div>', unsafe_allow_html=True)

    option = st.selectbox(
        "How would you like to be contacted?",
        ("Reduce Blur", "Reduce Noise", "Increase Light"),
        index=None,
        placeholder="Select contact method..."
    )
    st.write("You selected:", option)
    uploaded_file = st.file_uploader("Choose an image file to convert image", type=["png", "jpg", "jpeg"], key="uploader")
```

renoise : **เป็น** function **ที่เอาไว้ใช้ลด** noise **ในภาพ**

reblur : **เป็น** function **ที่เอาไว้ใช้ลดความ** blur **ในภาพ**

increaselight : **เป็น** function **ที่เอาไว้ใช้เพิ่มความสว่างให้ภาพ**

```python
def renoise(img):
    img_cv = np.array(img)
    img_med = cv.medianBlur(img_cv,5)
    img_out = reblur(img_med)
    return img_out

def reblur(img):
    kernel_sharpening = np.array([
        [-1, -1, -1],
        [-1, 9, -1],
        [-1, -1, -1]
    ])

    output_sharpened = cv.filter2D(img, -1, kernel_sharpening)
    return output_sharpened

def increaselight(img):
    img_cv = np.array(img)
    img_out = cv.convertScaleAbs(img_cv, alpha=1.2, beta=30)
    return img_out
```

convert_img_to bytes : **เป็น** fucntion **ที่แปลงการแสดงภาพแบบ** NumPy **เป็น** bytes **เพื่อสามารถให้**

```python
def convert_img_to_bytes(img_array):
    img_pil = Image.fromarray(img_array)
    img_bytes = BytesIO()
    img_pil.save(img_bytes, format='PNG')
    img_bytes.seek(0)
    return img_bytes
```

```python
if not np.array_equal(img_con, image_np):
    img_bytes = convert_img_to_bytes(img_con)
    st.download_button(label='Download Image', data=img_bytes, file_name="trans_image.png", mime="image/png")
else:
    st.write("No transformation was applied.")
```

User **ดาวน์โหลดภาพที่ผ่านการ ลด** noise , **ลดเบลอ หรือ เพิ่มแสง ได้**

About.py

เป็นหน้าที่ใช้แสดงรายชื่อผู้จัดทำ Project

```python
def show_about():
    load_css()
    st.markdown('<div style="margin:50px;"><h1 class="center-header">Team Members</h1></div>', unsafe_allow_html=True)
    img_nana = convert_image('src/Nana.jpg')
    img_pleum = convert_image('src/Pleum.jpg')
    img_phat = convert_image('src/Phat.jpg')
    img_ter = convert_image('src/Ter.jpg')
    st.markdown(
    f"""
    <div class="about-container">
        <div>
            <div class="circle-image">
                <img src="{img_nana}" alt="Nipatsa Chainiwattana">
            </div>
            <div class="name-text">Nipatsa Chainiwattana</div>
        </div>
        <div>
            <div class="circle-image">
                <img src="{img_ter}" alt="Puthipong Yomabut">
            </div>
            <div class="name-text">Puthipong Yomabut</div>
        </div>
        <div>
        <div>
            <div class="circle-image">
                <img src="{img_pleum}" alt="Patiharn Kamenkit">
            </div>
            <div class="name-text">Patiharn Kamenkit</div>
        </div>
        <div>
            <div class="circle-image">
                <img src="{img_phat}" alt="Phattaradanai Sornsawang">
            </div>
            <div class="name-text">Phattaradanai Sornsawang</div>
        </div>
    </div>
    """,
    unsafe_allow_html=True)

def load_css():
    with open('style.css') as f:
        st.markdown(f'<style>{f.read()}</style>', unsafe_allow_html=True)
```

Experiment results

1. Basic CNN Classifier

- ได้ค่า loss, accuracy, val_loss, val_accuracy ดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	loss	accuracy	val_loss	val_accuracy
0	0.694229	0.552852	0.656276	0.644960
1	0.592310	0.703279	0.530266	0.789164
2	0.515308	0.776656	0.488951	0.794548
3	0.490537	0.794369	0.542080	0.745920
4	0.473285	0.802441	0.502169	0.782938

ผลลัพธ์จากการใช้ 10 Epoch

	loss	accuracy	val_loss	val_accuracy
0	0.682937	0.578599	0.569515	0.740367
1	0.556697	0.751100	0.529145	0.780750
2	0.509908	0.780290	0.415575	0.833081
3	0.490191	0.793833	0.438066	0.814740
4	0.471593	0.802670	0.525559	0.783274
5	0.445410	0.818700	0.400871	0.841326
6	0.444493	0.815410	0.393058	0.849907
7	0.423921	0.824592	0.471738	0.794043
8	0.421032	0.827346	0.369239	0.858489
9	0.416182	0.831363	0.344781	0.866229

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลชุดทดสอบดังนี้

ผลลัพธ์จากการใช้ 5 epoch

	precision	recall	f1-score	support
0	0.93	0.75	0.83	4153
1	0.59	0.86	0.70	1781
accuracy			0.78	5934
macro avg	0.76	0.80	0.76	5934
weighted avg	0.83	0.78	0.79	5934

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.89	0.92	0.90	4153
1	0.80	0.73	0.76	1781
accuracy			0.86	5934
macro avg	0.84	0.83	0.83	5934
weighted avg	0.86	0.86	0.86	5934

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลภาพที่รวบรวมจากอินเตอร์เน็ตดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.53	0.85	0.65	20
1	0.62	0.25	0.36	20
accuracy			0.55	40
macro avg	0.58	0.55	0.51	40
weighted avg	0.58	0.55	0.51	40

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.54	1.00	0.70	20
1	1.00	0.15	0.26	20
accuracy			0.57	40
macro avg	0.77	0.57	0.48	40
weighted avg	0.77	0.57	0.48	40

2. ResNet50 Classifier

- ได้ค่า loss, accuracy, val_loss, val_accuracy ดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.3782561421394348	0.8469719290733337	0.43608126044273376	0.8189466595649719
2	0.331180602312088	0.8708825707435608	0.39231637120246887	0.8337540030479431
3	0.3202846348285675	0.8750143647193909	0.3274979591369629	0.8623591065406799
4	0.30998557806015015	0.8810971975326538	0.4336121678352356	0.8244994282722473
5	0.31656092405319214	0.8795669078826904	0.2789599895477295	0.8921419978141785

ผลลัพธ์จากการใช้ 10 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.37740468978881836	0.8466276526451111	0.28639376163482666	0.886252760887146
2	0.33068692684173584	0.871265172958374	0.38007333874702454	0.840484619140625
3	0.3257019817829132	0.8735988140106201	0.3211040496826172	0.8763250708580017
4	0.3206601142883301	0.877539336681366	0.26820695400238037	0.895002543926239
5	0.31194978952407837	0.8809441924095154	0.37059471011161804	0.8542823195457458
6	0.3174557685852051	0.8781896829605103	0.30611932277679443	0.8825508952140808
7	0.3121703565120697	0.881020724773407	0.276414155960083	0.9013965725898743
8	0.3264305889606476	0.877539336681366	0.2970423102378845	0.8896180391311646
9	0.3182157278060913	0.8793756365776062	0.3713380992412567	0.8500757217407227
10	0.3179555833339691	0.8791078329086304	0.2805759608745575	0.8965169191360474

-

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลชุดทดสอบดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.94	0.90	0.92	4153
1	0.78	0.86	0.82	1781
accuracy			0.89	5934
macro avg	0.86	0.88	0.87	5934
weighted avg	0.89	0.89	0.89	5934

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.93	0.92	0.93	4153
1	0.82	0.84	0.83	1781
accuracy			0.90	5934
macro avg	0.87	0.88	0.88	5934
weighted avg	0.90	0.90	0.90	5934

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลภาพที่รวบรวมจากอินเตอร์เน็ตดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.68	0.95	0.79	20
1	0.92	0.55	0.69	20
accuracy			0.75	40
macro avg	0.80	0.75	0.74	40
weighted avg	0.80	0.75	0.74	40

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.65	0.85	0.74	20
1	0.79	0.55	0.65	20
accuracy			0.70	40
macro avg	0.72	0.70	0.69	40
weighted avg	0.72	0.70	0.69	40

3. ResNet50 Fine-Tuned Classifier

- ได้ค่า loss, accuracy, val_loss, val_accuracy ดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.36693117022514343	0.852251410484314	0.3218940794467926	0.8729597926139832
2	0.30858364701271057	0.8785722255706787	0.2745767831802368	0.8970217108726501
3	0.2872086465358734	0.8902406096458435	0.2740139663219452	0.8963486552238464
4	0.2815428078174591	0.8930716514587402	0.22898237308502197	0.9101464152336121
5	0.27179333567619324	0.8937602639198303	0.24654550850391388	0.9052667021751404

ผลลัพธ์จากการใช้ 10 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.3765530288219452	0.8486552834510803	0.30561795830726624	0.8810365200042725
2	0.30609750747680664	0.882436215877533	0.3241766393184662	0.8818778395652771
3	0.2831873893737793	0.8922682404518127	0.2452692687511444	0.9056032299995422
4	0.27717718482017517	0.892650842666626	0.26119324564933777	0.8911324143409729
5	0.26885518431663513	0.896591305732727	0.2759481370449066	0.8923102617263794
6	0.2635817527770996	0.9006083011627197	0.2382548600435257	0.9094733595848083
7	0.2525058090686798	0.9016794562339783	0.2678695321083069	0.8894497752189636
8	0.24980802834033966	0.9035540819168091	0.22263766825199127	0.9098098874092102
9	0.24712461233139038	0.9038984179496765	0.23162464797496796	0.9057714939117432
10	0.24372754991054535	0.9060790538787842	0.25162920355796814	0.9013965725898743

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลชุดทดสอบดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.93	0.93	0.93	4153
1	0.85	0.85	0.85	1781
accuracy			0.91	5934
macro avg	0.89	0.89	0.89	5934
weighted avg	0.91	0.91	0.91	5934

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.95	0.90	0.93	4153
1	0.80	0.88	0.84	1781
accuracy			0.90	5934
macro avg	0.87	0.89	0.88	5934
weighted avg	0.90	0.90	0.90	5934

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลภาพที่รวบรวมจากอินเตอร์เน็ตดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.79	0.95	0.86	20
1	0.94	0.75	0.83	20
accuracy			0.85	40
macro avg	0.86	0.85	0.85	40
weighted avg	0.86	0.85	0.85	40

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.78	0.90	0.84	20
1	0.88	0.75	0.81	20
accuracy			0.82	40
macro avg	0.83	0.82	0.82	40
weighted avg	0.83	0.82	0.82	40

4. MoblienetV2 Fine-Tuned Classifier

- ได้ค่า loss, accuracy, val_loss, val_accuracy ดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.3629523813724518	0.8554267287254333	0.2968364655971527	0.8850748538970947
2	0.33972811698913574	0.8660622239112854	0.2942831516265869	0.8838970065116882
3	0.3204773962497711	0.8741726875305176	0.2803816497325897	0.8875988721847534
4	0.31725814938545227	0.8769654631614685	0.2903391718864441	0.8894497752189636
5	0.3093971014022827	0.8826274871826172	0.270304799079895	0.8939929604530334

ผลลัพธ์จากการใช้ 10 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.5539613366127014	0.7629595398902893	0.4149608910083771	0.8239946365356445
2	0.4354783296585083	0.8140709400177002	0.35650089383125305	0.8600033521652222
3	0.41878265142440796	0.8248593807220459	0.3989064395427704	0.8287060260772705
4	0.4039490818977356	0.8260453939437866	0.35016337037086487	0.8537775278091431
5	0.3896535038948059	0.8333907127380371	0.32422399520874023	0.8635369539260864
6	0.3871089518070221	0.834117591381073	0.3896649479866028	0.8243311643600464
7	0.37507960200309753	0.8401239514350891	0.34809839725494385	0.8578159213066101
8	0.3702645003795624	0.84222811460495	0.3285171687602997	0.8564698100090027
9	0.3696024715900421	0.8423811197280884	0.3420579433441162	0.855796754360199
10	0.36000391840934753	0.8500707745552063	0.3243740200996399	0.8687531352043152

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลชุดทดสอบดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.91	0.95	0.93	4153
1	0.88	0.77	0.82	1781
accuracy			0.90	5934
macro avg	0.89	0.86	0.87	5934
weighted avg	0.90	0.90	0.90	5934

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.94	0.89	0.91	4153
1	0.77	0.86	0.81	1781
accuracy			0.88	5934
macro avg	0.85	0.87	0.86	5934
weighted avg	0.89	0.88	0.88	5934

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลภาพที่รวบรวมจากอินเตอร์เน็ตดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.59	0.95	0.73	20
1	0.88	0.35	0.50	20
accuracy			0.65	40
macro avg	0.73	0.65	0.62	40
weighted avg	0.73	0.65	0.62	40

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.65	0.75	0.70	20
1	0.71	0.60	0.65	20
accuracy			0.68	40
macro avg	0.68	0.68	0.67	40
weighted avg	0.68	0.68	0.67	40

5. VGG19 Fine-Tuned Classifier

- ได้ค่า loss, accuracy, val_loss, val_accuracy ดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.5359932780265808	0.7613145112991333	0.4462161362171173	0.8191149234771729
2	0.4346888065338135	0.813917875289917	0.3824772834777832	0.831734836101532
3	0.4126732647418976	0.8256245255470276	0.43300971388816833	0.8103651404380798
4	0.40163087844848633	0.8294119834899902	0.3600488305091858	0.84502774477005
5	0.3936363458633423	0.8343088626861572	0.41211000084877014	0.8149082660675049

ผลลัพธ์จากการใช้ 10 Epoch

	loss	accuracy	val_loss	val_accuracy
1	0.5539613366127014	0.7629595398902893	0.4149608910083771	0.8239946365356445
2	0.4354783296585083	0.8140709400177002	0.35650089383125305	0.8600033521652222
3	0.41878265142440796	0.8248593807220459	0.3989064395427704	0.8287060260772705
4	0.4039490818977356	0.8260453939437866	0.35016337037086487	0.8537775278091431
5	0.3896535038948059	0.8333907127380371	0.32422399520874023	0.8635369539260864
6	0.3871089518070221	0.834117591381073	0.3896649479866028	0.8243311643600464
7	0.37507960200309753	0.8401239514350891	0.34809839725494385	0.8578159213066101
8	0.3702645003795624	0.84222811460495	0.3285171687602997	0.8564698100090027
9	0.3696024715900421	0.8423811197280884	0.3420579433441162	0.855796754360199
10	0.36000391840934753	0.8500707745552063	0.3243740200996399	0.8687531352043152

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลชุดทดสอบดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.93	0.82	0.87	4153
1	0.67	0.84	0.75	1781
accuracy			0.83	5934
macro avg	0.80	0.83	0.81	5934
weighted avg	0.85	0.83	0.83	5934

ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.90	0.92	0.91	4153
1	0.80	0.75	0.77	1781
accuracy			0.87	5934
macro avg	0.85	0.84	0.84	5934
weighted avg	0.87	0.87	0.87	5934

- ได้ค่า classification report เมื่อทดสอบกับข้อมูลภาพที่รวบรวมจากอินเตอร์เน็ตดังนี้

ผลลัพธ์จากการใช้ 5 Epoch

	precision	recall	f1-score	support
0	0.73	0.80	0.76	20
1	0.78	0.70	0.74	20
accuracy			0.75	40
macro avg	0.75	0.75	0.75	40
weighted avg	0.75	0.75	0.75	40

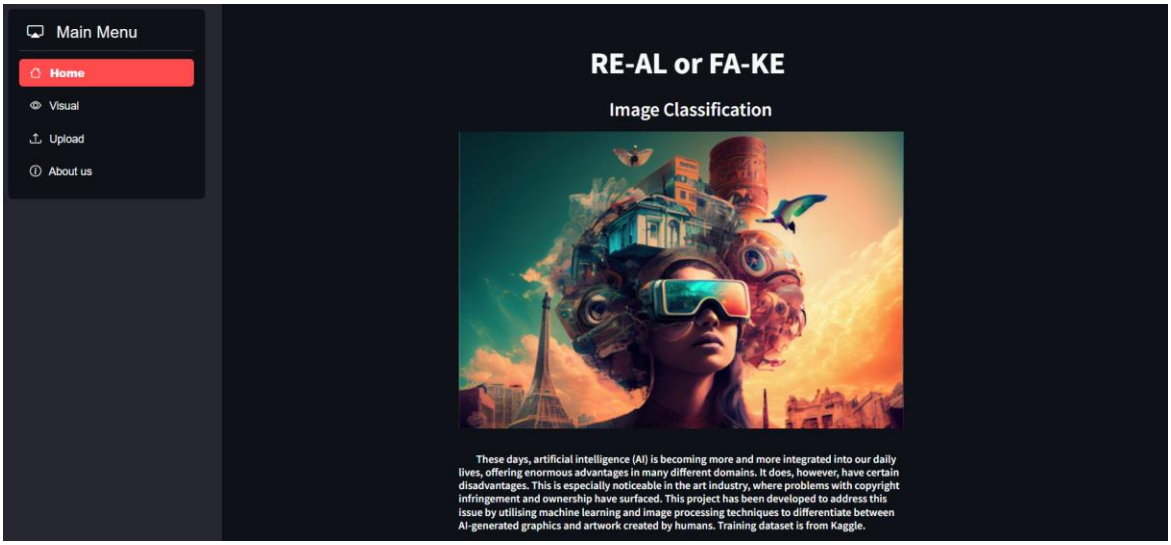
ผลลัพธ์จากการใช้ 10 Epoch

	precision	recall	f1-score	support
0	0.68	0.85	0.76	20
1	0.80	0.60	0.69	20
accuracy			0.72	40
macro avg	0.74	0.72	0.72	40
weighted avg	0.74	0.72	0.72	40

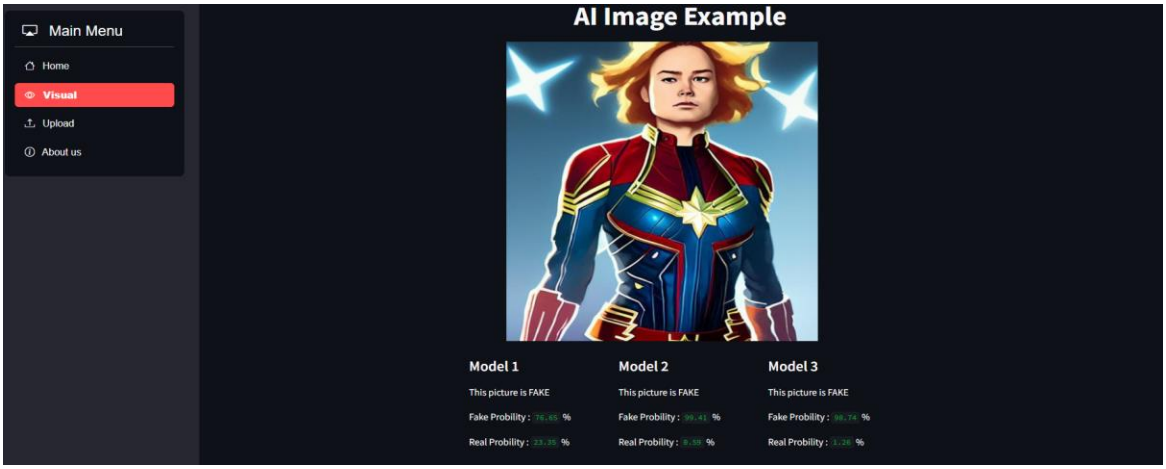
Web-Application Section

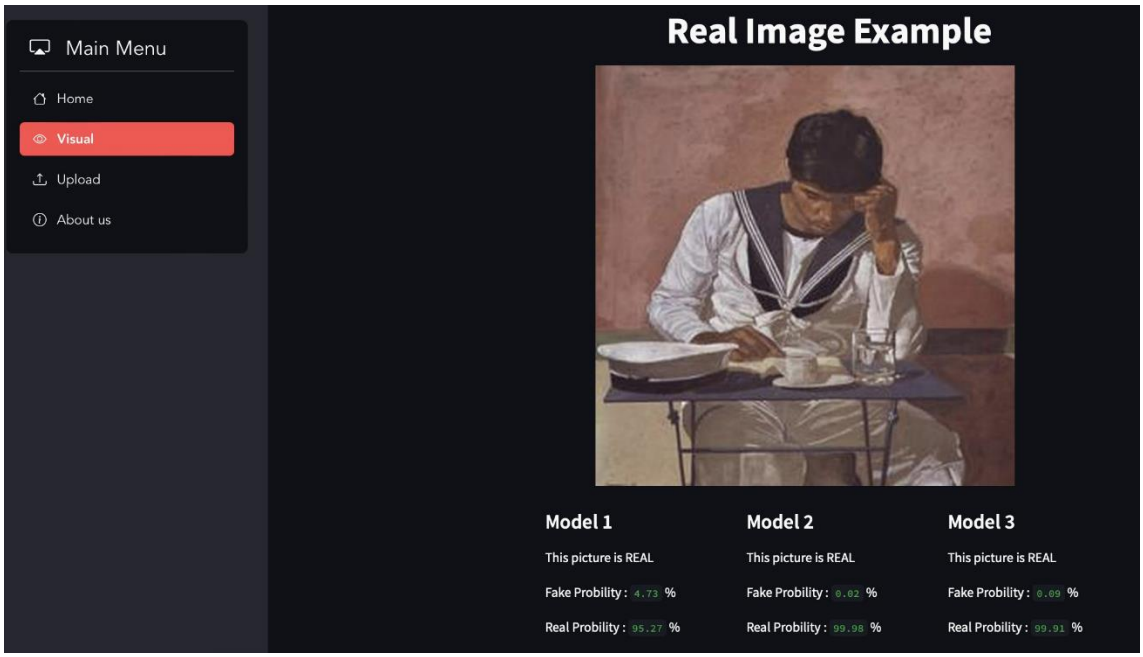
ได้เว็บแอพพลิเคชั่นที่แสดงผล 5 เมนูย่อยดังนี้

1. Home

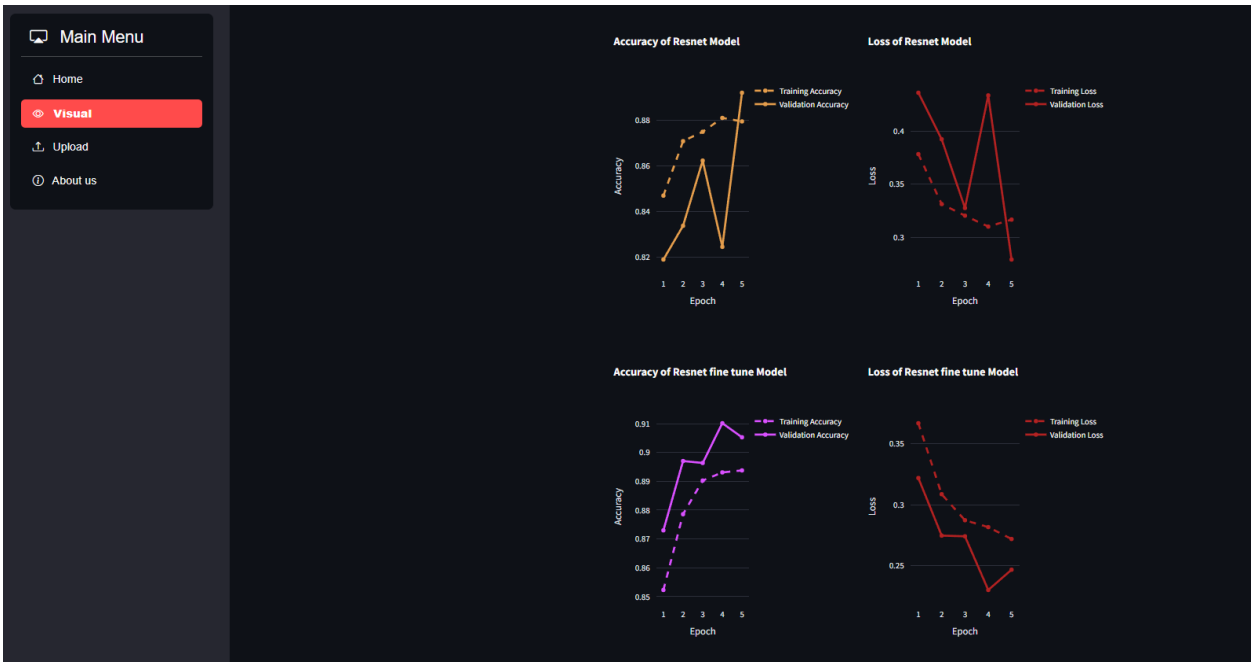


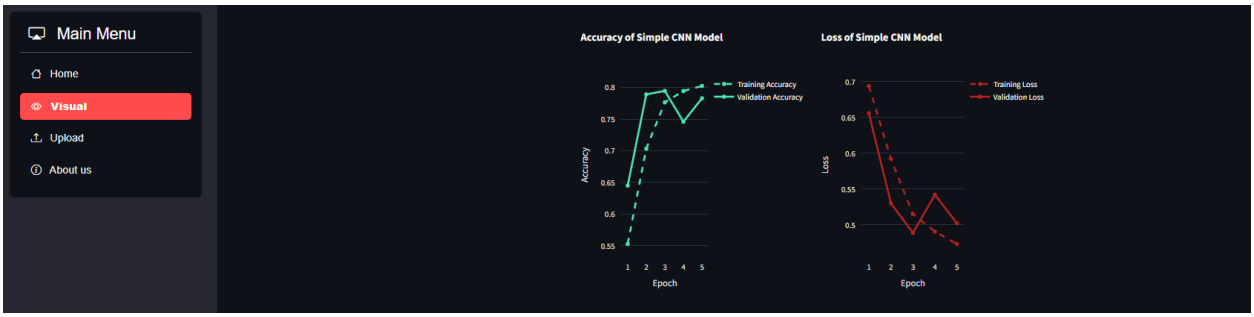
2. Visual


Main Menu
Home
Visual
Upload
About us
AI Image Example
Model 1
This picture is FAKE
Fake Probility : 76.65 %
Real Probility : 23.35 %
Model 2
This picture is FAKE
Fake Probility : 99.41 %
Real Probility : 0.59 %
Model 3
This picture is FAKE
Fake Probility : 98.74 %
Real Probility : 1.26 %


Main Menu
Home
Visual
Upload
About us
Real Image Example
Model 1
This picture is REAL
Fake Probility : 4.73 %
Real Probility : 95.27 %
Model 2
This picture is REAL
Fake Probility : 0.02 %
Real Probility : 99.98 %
Model 3
This picture is REAL
Fake Probility : 0.09 %
Real Probility : 99.91 %


Main Menu
Home
Visual
Upload
About us
Segment of AI and Real Image
AI painting pictures
REAL painting pictures
REAL painting pictures 44.5%
AI painting pictures 55.5%
Painting Pictures Count
18292
7947
4160
4153
Picture
Training
Testing
Validation
Category





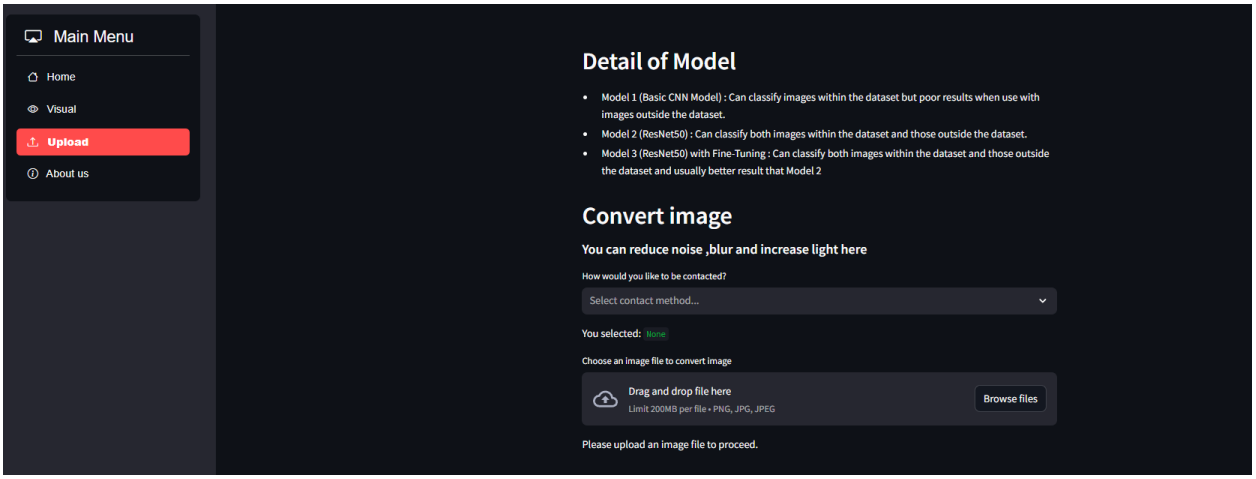
หมายเหตุ

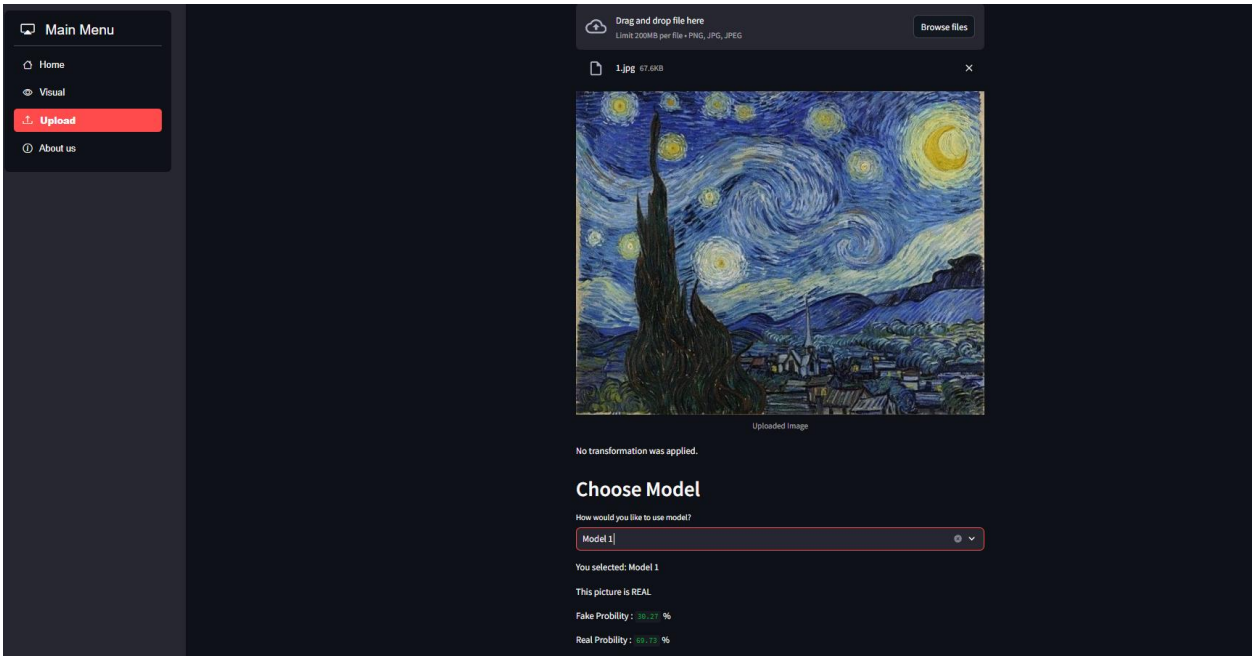
model1: Basic CNN Classifier

model2 : ResNet50 Classifier

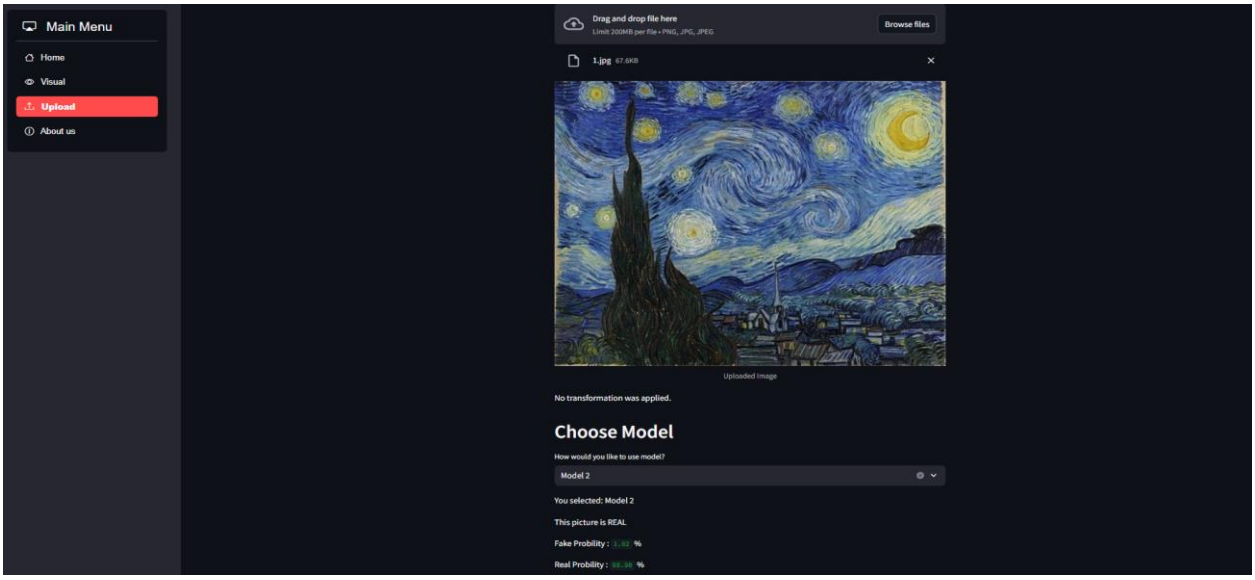
model3 ResNet50 Fine-Tuned Classifier

3.Upload

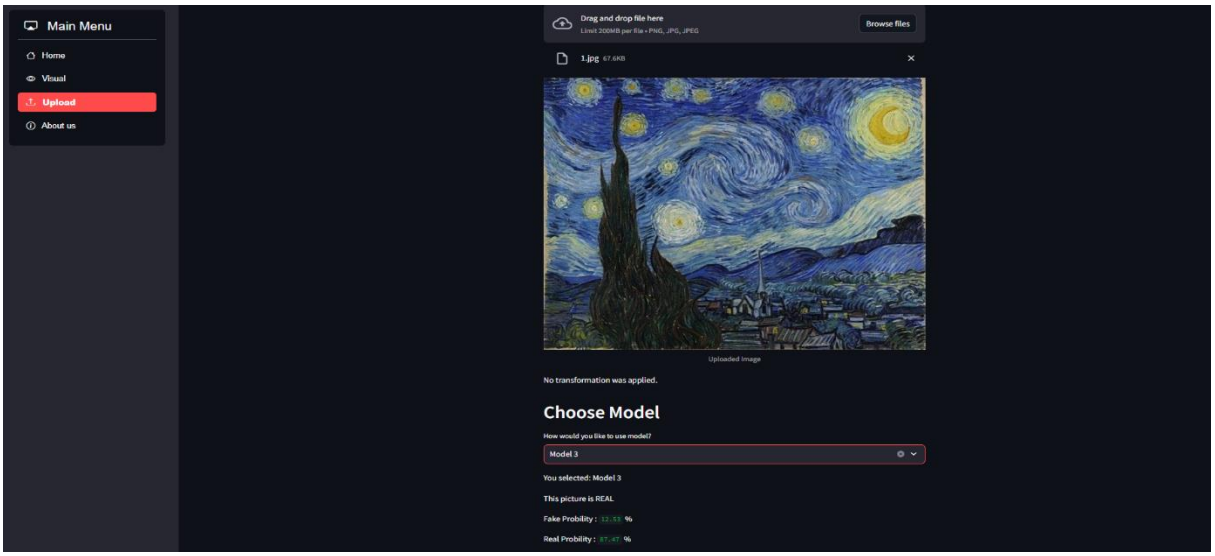




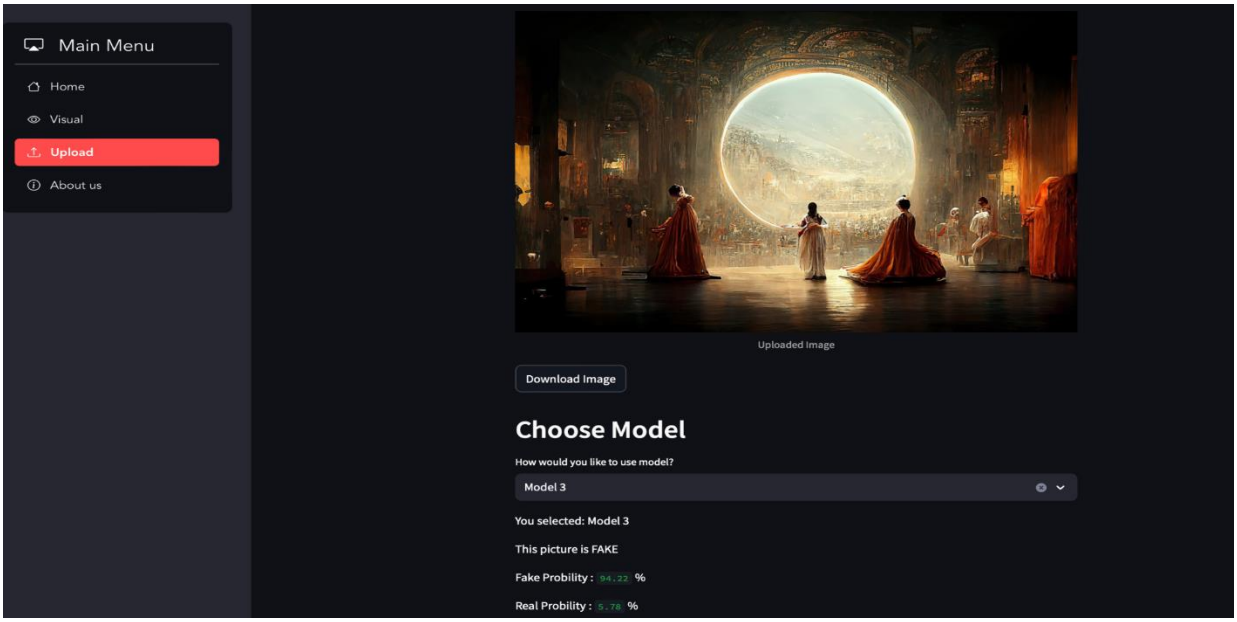
ภาพจากอินเตอร์เน็ตที่เป็นภาพจริง โดยเลือกใช้ model 1



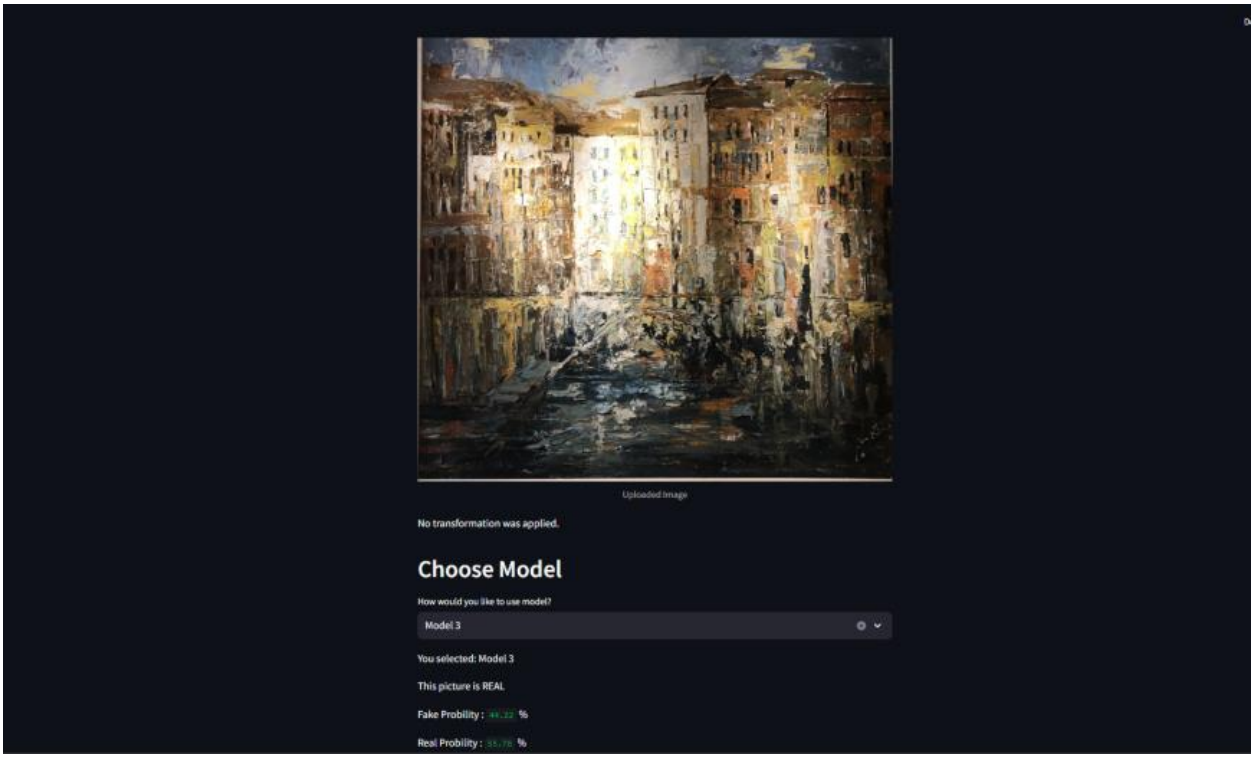
ภาพจากอินเตอร์เน็ตที่เป็นภาพจริง โดยเลือกใช้ model 2



ภาพจากอินเตอร์เน็ตที่เป็นภาพจริง โดยเลือกใช้ model 3



ภาพจากอินเตอร์เน็ตที่เป็นภาพ AI โดยเลือกใช้ model 3



ภาพจริงถ่ายจากกล้องโทรศัพท์ที่ ATT 19 โดย Reduce Noise และเลือกใช้ model 3

4. About us


Main Menu
Home
Visual
Upload
About us
Team Members
Nipatsa Chainiwattana
Puthipong Yomabut
Patiharn Kamenkit
Phattaradanai Sornsawang

Conclusion

จากการทดลองพบว่าในโมเดลตัวที่ 1 (Basic CNN Classifier) และ 4 (MobileNetV2 Classifier) เมื่อฝึกฝนด้วยจำนวน Epoch ที่ 10 จะให้ความแม่นยำสูงกว่าการฝึกฝนด้วยจำนวน 5 Epoch เมื่อทดสอบกับชุดข้อมูลใน Dataset และ ภาพที่ไม่อยู่ใน Dataset แต่ในโมเดลตัวที่ 2 (ResNet50 Classifier) โมเดลตัวที่ 3 (ResNet50 Fine-Tuned Classifier) และโมเดลตัวที่ 5 (VGG19 Fine-Tuned Classifier) การฝึกฝนด้วยจำนวน Epoch ที่ 10 จะให้ความแม่นยำสูงกว่าการฝึกฝนด้วยจำนวน 5 Epoch เมื่อทดสอบกับชุดข้อมูลใน Dataset แต่เมื่อทดสอบกับภาพที่ไม่อยู่ใน Dataset พบว่าความแม่นยำในการทำนายคลาสของภาพจริงลดลงเมื่อฝึกฝนด้วยจำนวน Epoch ที่มากขึ้น เนื่องจากทั้ง 2 คลาสมีลักษณะของภาพที่คล้ายกันมากซึ่งส่งผลให้โมเดลจดจำลักษณะเฉพาะของภาพใน Dataset มามากเกินไปส่งผลให้เมื่อเจอภาพที่ไม่เคยเห็นมาก่อนโมเดลจะไม่สามารถทำนายได้ โดยโมเดลที่ทำนายได้แม่นยำที่สุดคือโมเดลตัวที่ 3 (ResNet50 Fine-Tuned Classifier) รองลงมาคือโมเดลตัวที่ 5 (VGG19 Classifier) ตามด้วยโมเดลตัวที่ 2 (ResNet50 Classifier) โมเดลตัวที่ 4 (MobileNetV2 Classifier) และโมเดลตัวที่ 1 (Basic CNN Classifier) ตามลำดับ โดยผู้จัดทำได้เลือกใช้โมเดลตัวที่ 1 (Basic CNN Classifier) ที่ผ่านการฝึกฝนด้วยจำนวน 5 Epoch กับโมเดลตัวที่ 2 (ResNet50 Classifier) และโมเดลตัวที่ 3 (ResNet50 Fine-Tuned Classifier) ที่ผ่านการฝึกฝนด้วยจำนวน 10 Epoch สำหรับการใช้งานจริงในเว็บแอปพลิเคชันที่ได้พัฒนาขึ้น

Discussion and future work

เนื่องจากข้อจำกัดด้านทรัพยากรคอมพิวเตอร์ที่ยังไม่สามารถใช้งานในการเทรนโมเดลด้วย Dataset ขนาดใหญ่ได้ โมเดลปัจจุบันจึงสามารถจำแนกได้เฉพาะภาพวาดจริงที่มาจากมนุษย์และภาพวาดที่สร้างโดย Generative AI เท่านั้น โดยโปรเจคนี้ยังสามารถพัฒนาให้มีความแม่นยำในการจำแนกสูงขึ้นกว่าที่เป็นอยู่ในปัจจุบันหากมีความพร้อมของอุปกรณ์มากขึ้น ทางผู้จัดทำได้วางแผนที่จะพัฒนาโมเดลให้สามารถจำแนกภาพได้หลากหลายประเภทมากขึ้น โดยการเพิ่มจำนวนภาพใน Dataset ซึ่งอาจรวมถึงการจำแนกภาพถ่ายต่างๆและภาพวาดที่มาจากมนุษย์และจาก Generative AI รวมถึงมีการ Preprocess มากขึ้นเพื่อความแม่นยำ นอกจากนี้ยังมีแผนที่จะพัฒนาโมเดลให้สามารถใช้งานได้ในรูปแบบเว็บแอปพลิเคชันหรือแอปพลิเคชันที่สามารถให้ผู้ใช้บริการเข้ามาใช้งานได้

References

1. Zhu, M., Chen, H., Huang, M., Li, W., Hu, H., Hu, J., & Wang, Y. (2023, December 12). *GENDET: Towards good generalizations for AI-Generated Image Detection*. arXiv.org. https://arxiv.org/abs/2312.08880

2. Géron, A. (แปลโดย วิโรจน์ อัศวรังสี). (2024). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow: แนวคิด เครื่องมือ และเทคนิคสำหรับสร้างระบบอัจฉริยะ*. กรุงเทพฯ: Core Function.

3. Nogibjj. (n.d.). *GitHub - nogibjj/Detecting-AI-Generated-Fake-Images*. GitHub. https://github.com/nogibjj/Detecting-AI-Generated-Fake-Images