6-1

import os

import numpy as np

from tensorflow.keras.applications import VGG16

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.optimizers import Adam

from PIL import Image

import requests

from io import BytesIO

# Step 1: 加載預訓練的 VGG16 模型

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

# 添加自定義的分類層

x = base\_model.output

x = Flatten()(x)

x = Dense(128, activation='relu')(x)

predictions = Dense(2, activation='softmax')(x)

# 完整模型

model = Model(inputs=base\_model.input, outputs=predictions)

# 冷凍 VGG16 的預訓練層

for layer in base\_model.layers:

    layer.trainable = False

# 編譯模型

model.compile(optimizer=Adam(learning\_rate=0.0001), loss='categorical\_crossentropy', metrics=['accuracy'])

print("Model successfully built!")

# Step 2: 加載和準備本地數據集

# 設定數據集路徑

dataset\_path = os.path.join("Face-Mask-Detection", "dataset")

# 調整數據集路徑

data\_gen = ImageDataGenerator(rescale=1.0/255, validation\_split=0.2)

# 訓練數據生成器

train\_generator = data\_gen.flow\_from\_directory(

    dataset\_path,  # 本地數據集的主目錄

    target\_size=(224, 224),  # 圖片調整大小

    batch\_size=32,

    class\_mode='categorical',

    subset='training'  # 用於訓練的子集

)

# 驗證數據生成器

val\_generator = data\_gen.flow\_from\_directory(

    dataset\_path,  # 本地數據集的主目錄

    target\_size=(224, 224),

    batch\_size=32,

    class\_mode='categorical',

    subset='validation'  # 用於驗證的子集

)

print("Data successfully prepared!")

# Step 3: 訓練模型

history = model.fit(

    train\_generator,

    validation\_data=val\_generator,

    epochs=10,

    steps\_per\_epoch=train\_generator.samples // 32,

    validation\_steps=val\_generator.samples // 32

)

print("Model training complete!")

# Step 4: 定義 URL 圖片分類函數

def classify\_image(image\_url, model, class\_names):

    try:

        # 從 URL 加載圖片

        response = requests.get(image\_url)

        img = Image.open(BytesIO(response.content)).convert('RGB')

        img = img.resize((224, 224))

        img\_array = np.expand\_dims(np.array(img) / 255.0, axis=0)

        # 預測

        predictions = model.predict(img\_array)

        class\_idx = np.argmax(predictions, axis=1)[0]

        confidence = predictions[0][class\_idx]

        return class\_names[class\_idx], confidence

    except Exception as e:

        print(f"Error processing image: {e}")

        return None, None

# Step 5: 測試 URL 圖片分類

class\_names = ['With Mask', 'Without Mask']  # 修改為你模型的類別名稱

test\_image\_url = input("請輸入圖片 URL: ")

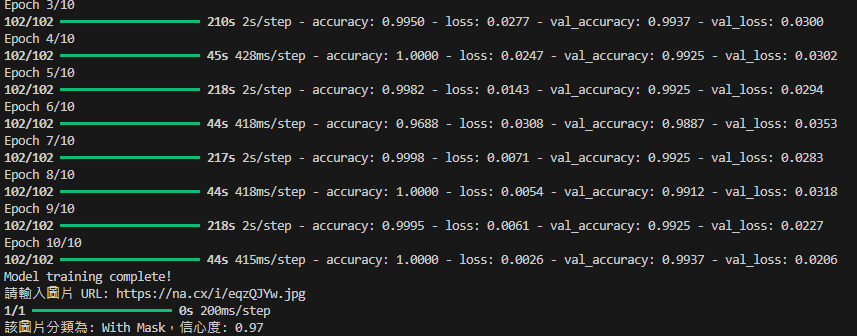
result, confidence = classify\_image(test\_image\_url, model, class\_names)

if result:

    print(f"該圖片分類為: {result}，信心度: {confidence:.2f}")

else:

    print("圖片分類失敗！")

6-2

from transformers import MarianMTModel, MarianTokenizer

from diffusers import StableDiffusionPipeline

import matplotlib.pyplot as plt

import torch

def translate\_text(input\_text):

    """翻譯中文為英文"""

    model\_name = "Helsinki-NLP/opus-mt-zh-en"

    tokenizer = MarianTokenizer.from\_pretrained(model\_name)

    model = MarianMTModel.from\_pretrained(model\_name)

    # 使用模型進行翻譯

    translated = model.generate(\*\*tokenizer([input\_text], return\_tensors="pt", padding=True))

    translation = [tokenizer.decode(t, skip\_special\_tokens=True) for t in translated][0]

    return translation

def generate\_image(description):

    """根據英文描述生成圖片"""

    # 加載 Stable Diffusion 模型

    sd\_pipeline = StableDiffusionPipeline.from\_pretrained("runwayml/stable-diffusion-v1-5")

    sd\_pipeline.to("cuda" if torch.cuda.is\_available() else "cpu")

    # 使用模型生成圖片

    image = sd\_pipeline(description).images[0]

    return image

def display\_image(image, title):

    """使用 Matplotlib 顯示圖片"""

    plt.figure(figsize=(8, 8))

    plt.imshow(image)

    plt.axis("off")

    plt.title(title, fontsize=16)

    plt.show()

def main():

    while True:

        # 提示使用者輸入中文描述

        user\_input = input("請輸入中文描述（或輸入 'exit' 結束）：")

        if user\_input.lower() == "exit":

            print("程式結束。")

            break

        # 步驟 1: 翻譯中文

        print("翻譯中...")

        translated\_text = translate\_text(user\_input)

        print(f"翻譯結果：{translated\_text}")

        # 步驟 2: 生成圖片

        print("生成圖片中，請稍候...")

        image = generate\_image(translated\_text)

        # 步驟 3: 使用 Matplotlib 顯示圖片

        print("顯示圖片...")

        display\_image(image, title=user\_input)

        # 儲存圖片

        image.save("generated\_image.png")

        print("圖片已儲存為 'generated\_image.png'")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

