HW6-1:Transfer learning on pretrained VGG 16 (see Lecture 17)

| import tensorflow as tf  from tensorflow.keras.applications import VGG16  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense, Flatten, Dropout  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras.optimizers import Adam  import numpy as np  import os  import random  import matplotlib.pyplot as plt  from tensorflow.keras.preprocessing import image  # =========================================  # Step 1: 建立 VGG16 預訓練模型  # =========================================  base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))  model = Sequential([  base\_model,  Flatten(),  Dense(256, activation='relu'),  Dropout(0.5),  Dense(2, activation='softmax') # 修改為 2 個類別  ])  base\_model.trainable = False # 凍結預訓練層  model.compile(optimizer=Adam(learning\_rate=0.0001),  loss='categorical\_crossentropy',  metrics=['accuracy'])  model.summary()  # =========================================  # Step 2: 複製並預處理資料集  # =========================================  # 下載公開的口罩數據集  !git clone https://github.com/prajnasb/observations.git  os.makedirs("dataset/train/mask", exist\_ok=True)  os.makedirs("dataset/train/no\_mask", exist\_ok=True)  import shutil  mask\_path = "observations/experiements/data/with\_mask"  no\_mask\_path = "observations/experiements/data/without\_mask"  for img in os.listdir(mask\_path):  shutil.move(os.path.join(mask\_path, img), "dataset/train/mask")  for img in os.listdir(no\_mask\_path):  shutil.move(os.path.join(no\_mask\_path, img), "dataset/train/no\_mask")  # 使用 ImageDataGenerator 處理數據  train\_datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)  train\_generator = train\_datagen.flow\_from\_directory(  'dataset/train',  target\_size=(224, 224),  batch\_size=32,  class\_mode='categorical',  subset='training')  val\_generator = train\_datagen.flow\_from\_directory(  'dataset/train',  target\_size=(224, 224),  batch\_size=32,  class\_mode='categorical',  subset='validation')  # =========================================  # Step 3: 訓練模型  # =========================================  history = model.fit(train\_generator,  validation\_data=val\_generator,  epochs=5)  # =========================================  # Step 4: 從資料集中隨機選擇10張圖片並分類  # =========================================  def predict\_and\_show\_random\_images(model, dataset\_path, class\_names, num\_images=10):  all\_images = []  for root, \_, files in os.walk(dataset\_path):  for file in files:  if file.endswith(('jpg', 'jpeg', 'png')):  all\_images.append(os.path.join(root, file))    # 隨機選擇 num\_images 張圖片  selected\_images = random.sample(all\_images, num\_images)    plt.figure(figsize=(15, 10))  for i, img\_path in enumerate(selected\_images):  # 預處理圖片  img = image.load\_img(img\_path, target\_size=(224, 224))  img\_array = image.img\_to\_array(img) / 255.0  img\_array = np.expand\_dims(img\_array, axis=0)  # 預測類別  predictions = model.predict(img\_array)  predicted\_class = class\_names[np.argmax(predictions)]  confidence = np.max(predictions) \* 100  # 顯示圖片與預測結果  plt.subplot(2, 5, i+1)  plt.imshow(img)  plt.title(f"{predicted\_class}\n{confidence:.2f}%")  plt.axis('off')  plt.tight\_layout()  plt.show()  # 測試: 從訓練資料夾中選10張圖片  class\_names = list(train\_generator.class\_indices.keys()) # 獲取類別名稱  predict\_and\_show\_random\_images(model, "dataset/train", class\_names, num\_images=10) |
| --- |



HW6-2: Hugging Face- AI 應用

安裝

| !pip install diffusers transformers accelerate scipy |
| --- |

去 Hugging Face 註冊 取得 Hugging Face Token

| from huggingface\_hub import login  # 輸入你的 Hugging Face Token  login("hf\_DODMfPmUFhytvkHJzRYRrhUGvvdPKUaKtp") |
| --- |

請他依照文字生成圖片，使用Stable Diffusion

| from diffusers import StableDiffusionPipeline  import torch  from PIL import Image  from IPython.display import display # 用於在 Colab 中顯示圖片  # 載入模型  pipe = StableDiffusionPipeline.from\_pretrained("runwayml/stable-diffusion-v1-5", torch\_dtype=torch.float16)  pipe = pipe.to("cuda") # 使用 GPU 加速  # 自定義函數來生成圖片  def generate\_image(prompt):  """  根據輸入的文字描述生成影像，並在 Colab 中顯示。  :param prompt: 用於描述影像的文字。  """  print(f"Generating image for: {prompt}")  image = pipe(prompt).images[0] # 生成影像  display(image) # 在 Colab 中顯示圖片  return image  # 輸入文字並生成影像  text\_prompt = input("請輸入你的文字描述: ")  image = generate\_image(text\_prompt) |
| --- |