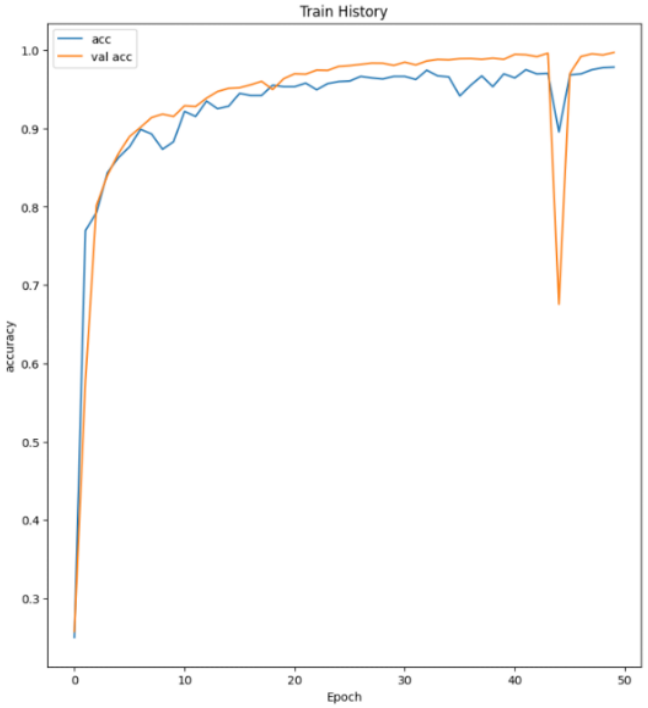
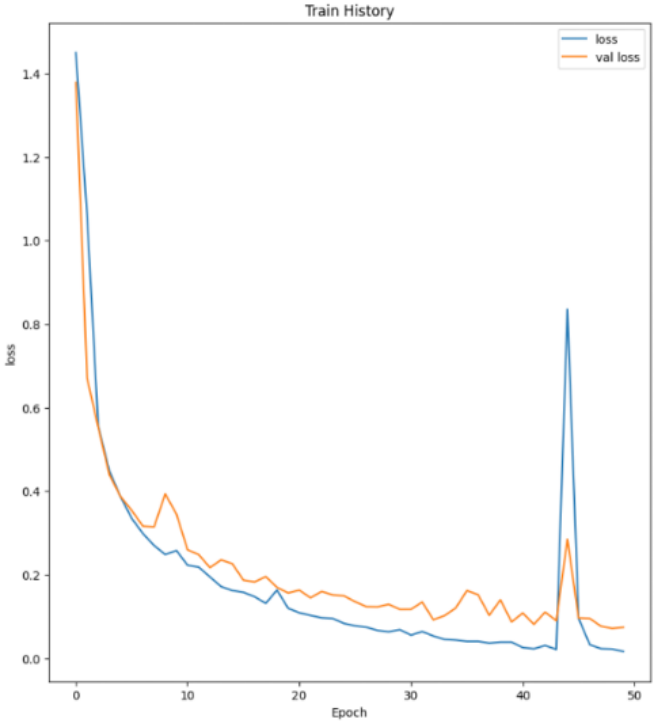
MidTerm Project - 工件辨識期中專題
109504501曾千芸

一、工件辨識準確率及損失率

1. Train accuracy history圖



2. Train loss history圖



3. Test accuracy, Test loss 截圖

```
test accuracy 0.9783464670181274
test loss 0.07384157180786133
```

二、程式碼解釋

1. 導入

```
導入

import tensorflow as tf
import os
import numpy as np
from matplotlib import pyplot as plt
from tensorflow.keras.layers import Conv2D, BatchNormalization, Activation, MaxPool2D, Dropout, Flatten, Dense
from tensorflow.keras import Model
import cv2
```

2. 導入資料

```
導入資料

dic={'bolt': 0, 'locatingpin': 1, 'nut': 2, 'washer': 3}
```

```python
X=[] # 建立空的X列表，用來儲存圖片和對應的標籤
Y=[] # 建立空的Y列表，用來儲存圖片和對應的標籤
# 定義名為read_d的函數，該函數接受一個目錄名稱作為輸入
def read_d(directory_name):
    global X,Y # 使用global關鍵字來存取在函數外定義的X和Y列表
    # 遍歷 directory_name 目錄中的內容。os.listdir 函數返回指定目錄中的文件名列表
    for i in os.listdir(directory_name):
        # 對文件夾中每個文件進行遍歷
        for filename in os.listdir(directory_name+'\\'+i):
            # 讀取圖片
            img = cv2.imread(directory_name+'\\'+i + "\\" + filename,cv2.IMREAD_GRAYSCALE)
            if not img is None :
                # 將所有文件變為同一大小
                img=cv2.resize(img,(128,128),interpolation=cv2.INTER_CUBIC)
                # 改格式使得可以卷積計算
                img=img.astype('float64')
                Y.append(dic[i])
                # 將圖片加入訓練集
                X.append(img)
    return X,Y
read_d('MidTerm_Dataset\\train\\')
# 使用NumPy將X和Y轉換為NumPy數組形式，並將X重塑為形狀為(len(X), 128, 128, 1)的四維數組，以便與卷積神經網絡進行訓練
X_train=np.array(X).reshape(len(X),128,128,1)
Y_train=np.array(Y)
```

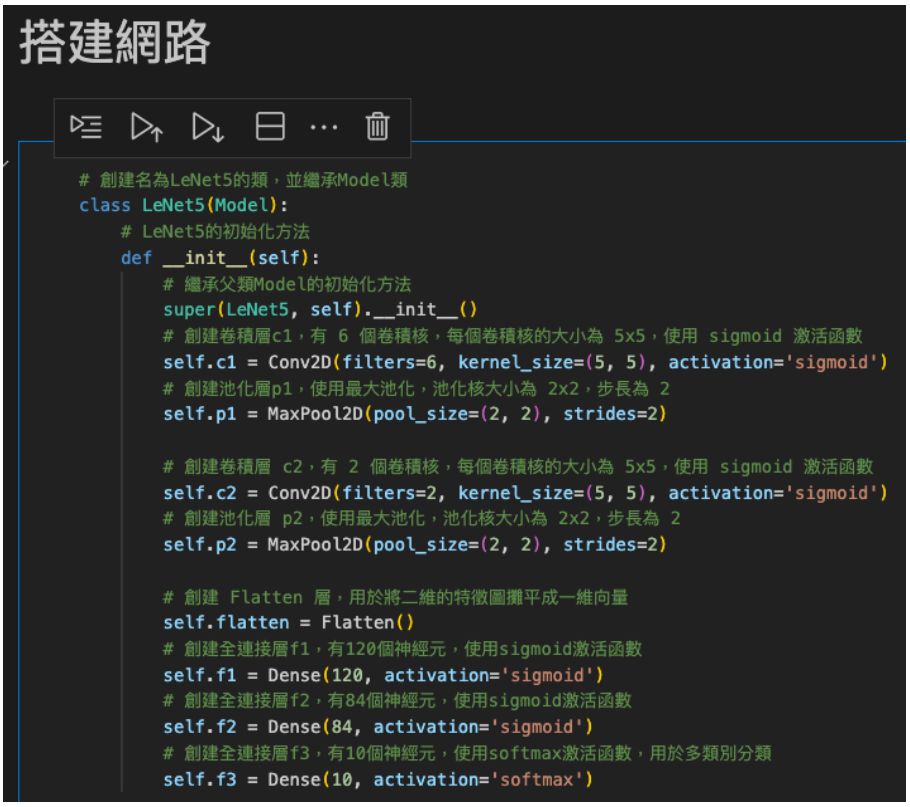
```python
# 創建空列表X和Y，以存儲圖片和相應的標籤
X=[]
Y=[]
# 定義一個函數read_d，它接受一個目錄名稱作為輸入
def read_d(directory_name):
    # 使用global關鍵字訪問在函數外部定義的X和Y列表
    global X,Y
    # 循環遍歷directory_name目錄中的內容。os.listdir函數返回指定目錄中的文件名列表
    for i in os.listdir(directory_name):
        # 對文件夾中每個文件進行遍歷
        for filename in os.listdir(directory_name+'\\'+i):
            # 讀取圖片
            img = cv2.imread(directory_name+'\\'+i + "\\" + filename,cv2.IMREAD_GRAYSCALE)
            if not img is None :
                # 將所有文件變為同一大小
                img=cv2.resize(img,(128,128),interpolation=cv2.INTER_CUBIC)
                # 改格式使得可以卷積計算
                img=img.astype('float64')
                # 將對應的標籤添加到Y列表
                Y.append(dic[i])
                # 將圖片加入訓練集
                X.append(img)
    return X,Y
# 調用read_d函數，從測試目錄讀取測試圖片
read_d('MidTerm_Dataset\\test\\')
# 使用NumPy將X和Y轉換為NumPy數組形式，並將X重塑為形狀為(len(X), 128, 128, 1)的四維數組，以便與卷積神經網絡進行訓練
X_test=np.array(X).reshape(len(X),128,128,1)
Y_test=np.array(Y)
```

3. 圖片大小

```python
import random
```

```
# 設定隨機數種子為 7，確保每次隨機產生的結果都一樣
np.random.seed(7)
# 對 X_train 進行隨機排序，打亂訓練集中的圖片順序
np.random.shuffle(X_train)
# 再次設定隨機數種子為 7，確保和之前相同的隨機數序列
np.random.seed(7)
# 對 Y_train 進行隨機排序，打亂訓練集中的標籤順序，確保圖片與標籤的對應關係不變
np.random.shuffle(Y_train)
# 設定 TensorFlow 中的隨機數種子為 7，確保 TensorFlow 中的隨機結果和 numpy 中的相同
tf.random.set_seed(7)
```

4. 搭建網路

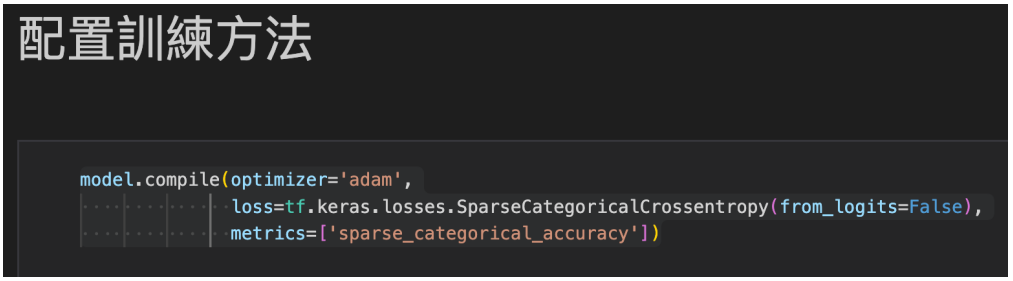



```
    # 定義 LeNet5 的呼叫方法
    def call(self, x):
        x = self.c1(x) # 將輸入x經過卷積層c1轉換
        x = self.p1(x) # 將經過卷積層c1的輸出x進行最大池化，得到p1的輸出

        x = self.c2(x) # 將p1的輸出x經過卷積層c2轉換
        x = self.p2(x) # 將經過卷積層c2的輸出x進行最大池化，得到p
        x = self.flatten(x) # 將feature maps轉化為向量
        x = self.f1(x) # 第一個全連接層，有120個神經元，使用sigmoid作為激活函數
        x = self.f2(x) # 第二個全連接層，有84個神經元，使用sigmoid作為激活函數
        y = self.f3(x) # 輸出層，有10個神經元，使用softmax作為激活函數，用於將輸出轉化為概率分佈
        return y

# 創建一個LeNet5模型，並將其賦值給變量model
model = LeNet5()
```

5. 配置訓練方法



使用了 compile 方法對模型進行編譯，設定了優化器、損失函數和評價指標

optimizer='adam'：選擇 Adam 優化器作為優化器，用於根據損失函數的梯度更新模型參數
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False)：選擇稀疏分類交叉熵作為損失函數
from_logits=False 表示模型輸出的是經過 softmax 函數歸一化後的結果，而不是經過線性轉換的結果
metrics=['sparse_categorical_accuracy']：選擇稀疏分類精度作為評價指標，用於評估模型的性能

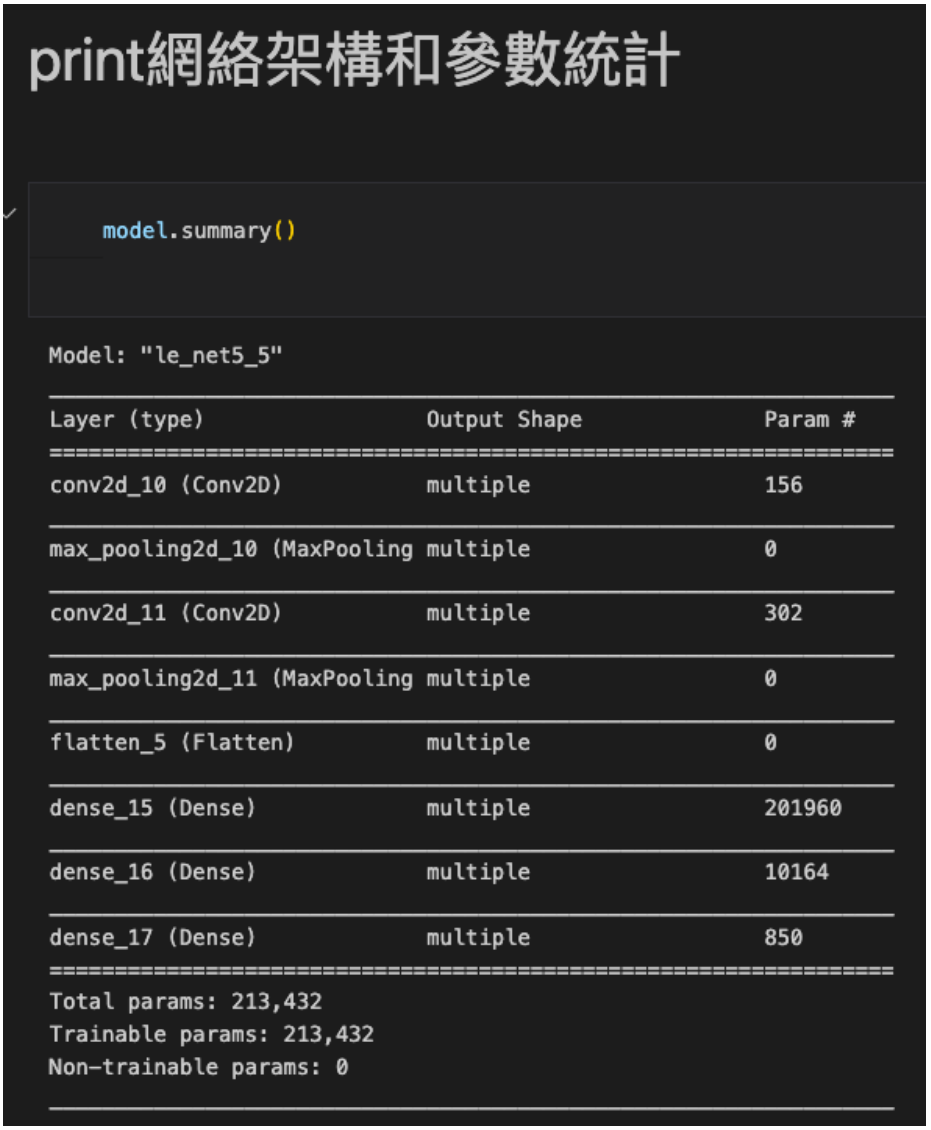
6. 模型訓練


```
history =model.fit(X_train, Y_train, batch_size=10, epochs=10, validation_data=(X_test, Y_test), validation_freq=1)
```

history是用來存儲模型在訓練過程中的一些重要參數和指標，如訓練集和驗證集的損失和準確率等

model.fit()方法用來訓練模型：
X_train: 訓練集的輸入特徵
Y_train: 訓練集的輸出標籤
batch_size: 訓練過程中每一批次的樣本數量
epochs: 訓練的輪數
validation_data: 用於驗證的數據，包括輸入特徵和輸出標籤
validation_freq: 驗證的頻率，即每訓練幾輪進行一次驗證

7. print網路架構和參數統計



8. 寫入參數

9. 顯示訓練集和驗證集的accuracy和loss曲線

```
from matplotlib.pyplot import MultipleLocator
```

```
# 從history中取出訓練過程中的sparse_categorical_accuracy(準確率)值，並將其存儲在acc中
acc = history.history['sparse_categorical_accuracy']
# 從history中取出訓練過程中的validation_sparse_categorical_accuracy(驗證準確率)值，並將其存儲在val_acc中
val_acc = history.history['val_sparse_categorical_accuracy']
# 從history中取出訓練過程中的loss(損失)值，並將其存儲在loss中
loss = history.history['loss']
# 從history中取出訓練過程中的validation_loss(驗證損失)值，並將其存儲在val_loss中
val_loss = history.history['val_loss']


plt.figure(figsize=(20,10))
# 定位子圖，在一行中創建一個包含兩個子圖的畫布，目前定位於第一個子圖
plt.subplot(1, 2, 1)
y=MultipleLocator(10)
ax=plt.gca()
ax.xaxis.set_major_locator(y)
# 繪製val_acc(驗證準確率)的折線圖，並給圖例標籤命名為"acc"
plt.plot(val_acc, label='acc')
# 繪製acc(訓練準確率)的折線圖，並給圖例標籤命名為"val acc"
plt.plot(acc, label='val acc')
plt.xlabel('Epoch')

plt.ylabel('accuracy')
# 設置子圖標題為"Train History"
plt.title('Train History')
# 創建圖例
plt.legend()

# 定位子圖，在一行中創建一個包含兩個子圖的畫布，目前定位於第二個子圖
plt.subplot(1, 2, 2)
y=MultipleLocator(10)
ax=plt.gca()
ax.xaxis.set_major_locator(y)
# 繪製 loss(訓練損失)的折線圖，並給圖例標籤命名為"loss"
plt.plot(loss, label='loss')
# 繪製val_loss(驗證損失)的折線圖，並給圖例標籤命名為"val loss"
plt.plot(val_loss, label='val loss')
# 設置子圖標題為"Train History"
plt.xlabel('Epoch')
plt.ylabel('loss')
plt.title('Train History')
# 創建圖例
plt.legend()
# 顯示繪製的圖片
plt.show()
```