主題描述：

主題涵蓋了使用TensorFlow和Keras建立卷積神經網路(CNN)來處理MNIST手寫數字資料集的完整流程。首先，載入資料集並進行預處理，包括歸一化和標籤的One-hot編碼，再來利用Sequential模型建構了包含多個卷積層和全連接層的神經網路結構。模型經過編譯，使用Adam優化器和分類交叉熵損失函數進行訓練，同時監控訓練過程中的準確率。訓練完成後，評估模型在測試集上的表現，並繪製了訓練和驗證誤差隨時間的變化圖。

引用資料與程式碼敘述：

1、MNIST：針對手寫數字資料集進行處理和分類的。

程式碼：

from tensorflow.keras.datasets import mnist

# 載入資料集

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

2、CNN：使用卷積神經網絡（CNN）模型來處理MNIST資料集的。

程式碼：

# 模型架構／調整模型結構

model = models.Sequential([

layers.Conv2D(16, (3, 3), activation='relu', padding='same', input\_shape=(28, 28, 1)),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(32, (3, 3), activation='relu', padding='same'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu', padding='same'),

layers.MaxPooling2D((2, 2)),

layers.Flatten(),

layers.Dense(256, activation='relu'),

layers.Dense(10, activation='softmax')

])

3、Tuned：這個程式是對模型的超參數進行了調優，以優化模型的性能和準確率。

程式碼：

# 訓練模型／超參數調優

history = model.fit(X\_train, y\_train, epochs=10, batch\_size=128, validation\_data=(X\_val, y\_val))

自行部份修改對比：

1. 深度學習架構的選擇：

原始程式：使用了自訂的深度學習框架，包括手動實作神經網路的層和優化器。

修改後的程式：使用了TensorFlow和Keras作為深度學習框架，利用其內建的高階API來建置、編譯和訓練模型。

1. 模型建置與編譯：

原始程式：模型的建構是透過逐層添加自訂的神經網路層完成的，例如手動實現的捲積層、池化層和全連接層。

修改後的程式：利用Keras的Sequential模型和內建的層類別來更簡潔地定義卷積神經網路結構，並透過compile方法指定最佳化器和損失函數。

1. 資料預處理與分割：

原始程式：資料載入後進行了自訂的歸一化處理和One-hot編碼。

修改後的程式：使用sklearn提供的LabelBinarizer進行標籤的One-hot編碼，並且利用TensorFlow的內建函數進行資料的歸一化和形狀調整，使得資料預處理步驟更加簡潔和規範化。

1. 訓練與評估過程：

原始程式：訓練過程使用了自訂的訓練函數，並且繪製了自訂的訓練和驗證誤差圖。

修改後的程式：使用了Keras的fit方法進行模型訓練，並利用內建的evaluate方法評估模型效能，同時利用Matplotlib繪製了訓練和驗證誤差隨時間的變化圖。

心得：

這次從自定義深度學習框架轉換到TensorFlow和Keras的過程讓我深刻體會到現代深度學習工具的便利性和效率。TensorFlow和Keras提供的內置功能和高級API大幅簡化模型的構建、訓練和評估過程，同時保持代碼的清晰度和可讀性。使用內置的優化器、损失函数和評估指標，不僅提高了開發速度，還有效地避免了潛在的錯誤。另外標準化的數據處理流程使得整體開發過程更加順暢和符合行業標準。