**第二周作業部分為ChatGPT 4o mini協助完成**

主要是我想提高模型準確率請他提供幫助，原始模型評估後的準確度為:

Test loss: 0.11853646486997604

Test accuracy: 0.977400004863739

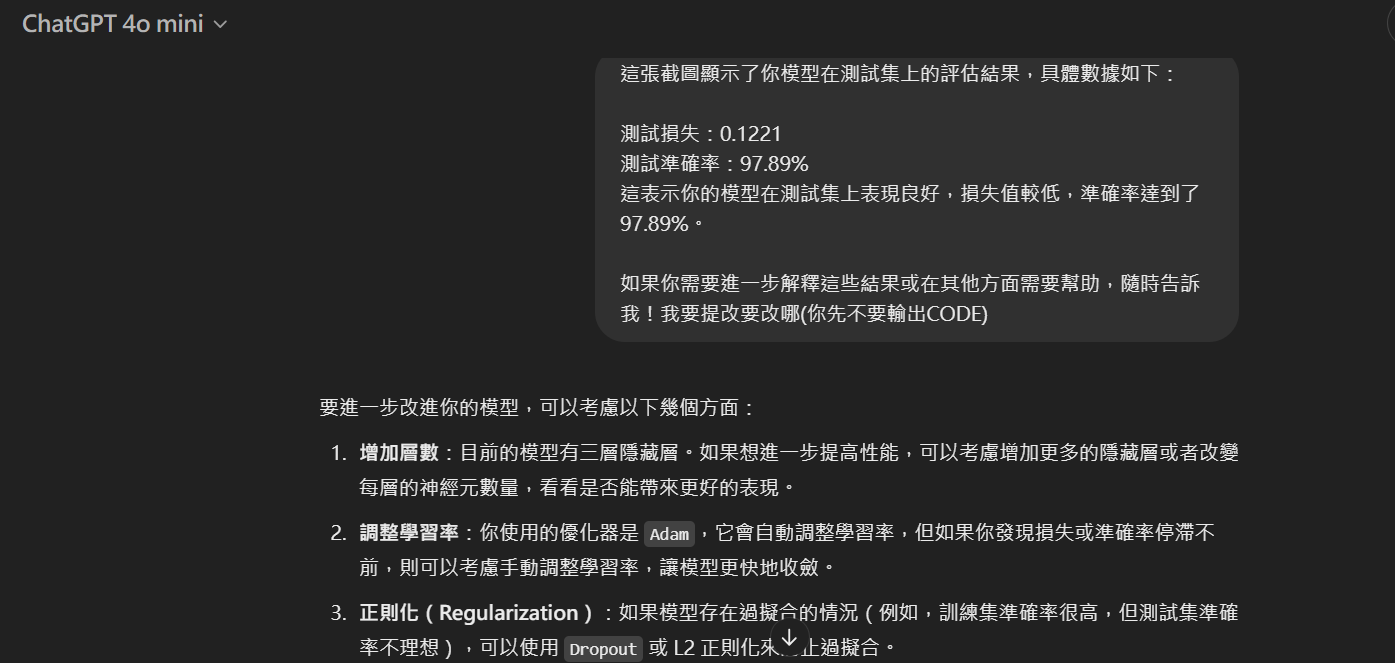
我請ChatGPT 4o mini協助我提高準確度，他給我幾個選擇，其中他提到如果我發現目前的模型在訓練集和測試集之間有顯著的準確度差異（例如，訓練集準確率高，測試集準確率低），那麼 正則化（Dropout 或 L2 regularization） 是比較有效的選擇。而L2 正則化會在損失函數中加入權重的平方和，這樣有助於防止模型的權重過大，從而減少過擬合。

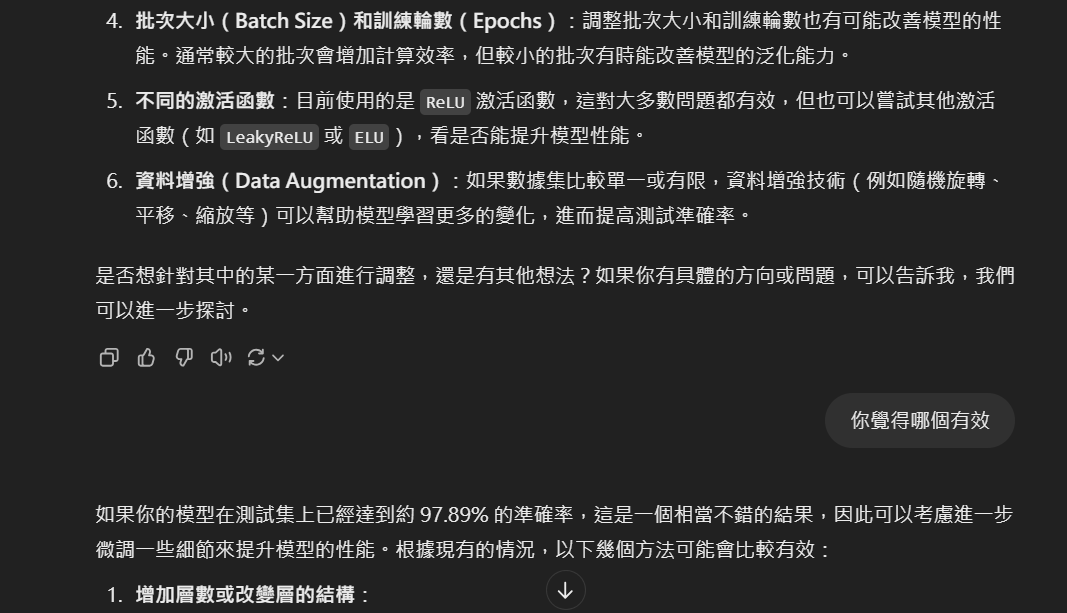
於是我選擇請他生成使用 L2 正則化的代碼來調整模型，測試後的準確度為:

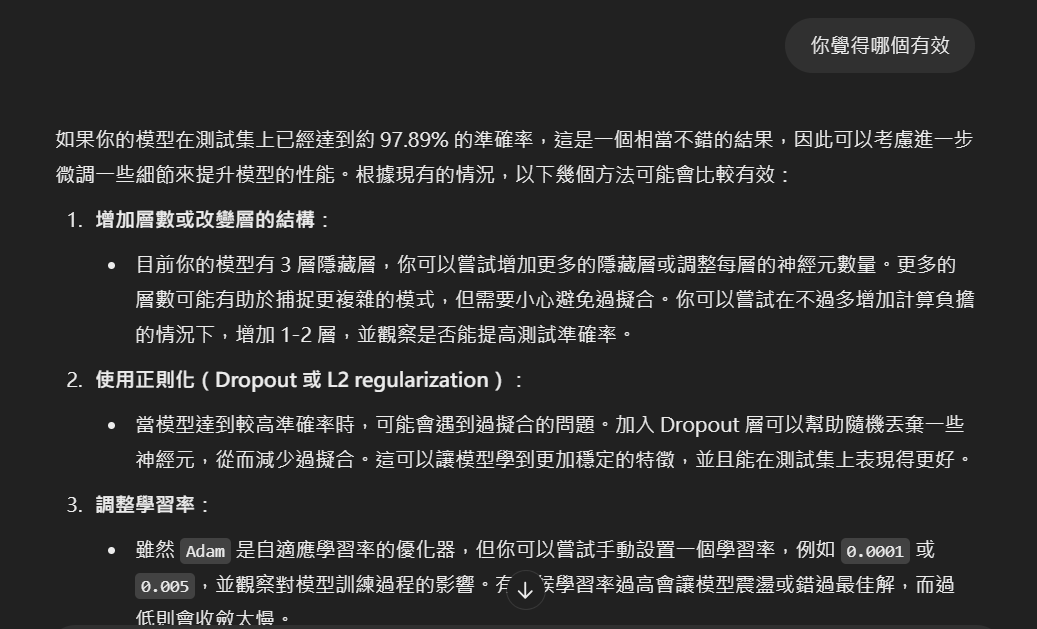
Test loss: 0.3739358186721802

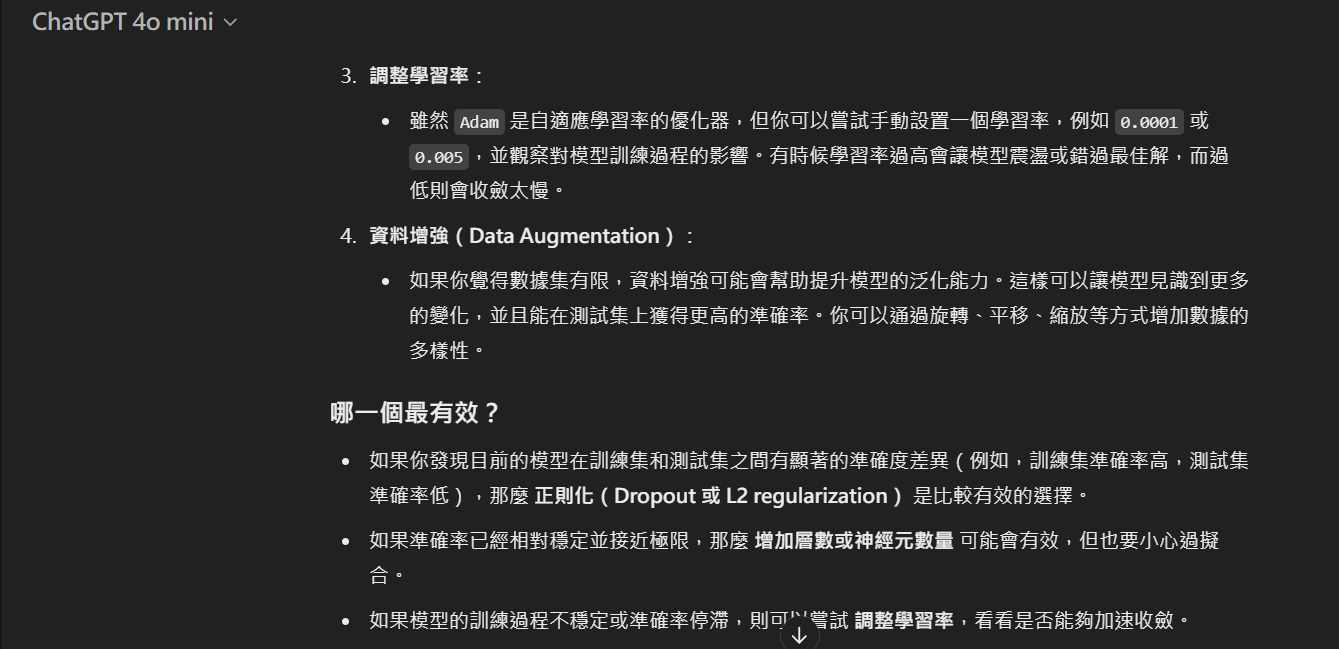
Test accuracy: 0.951200008392334

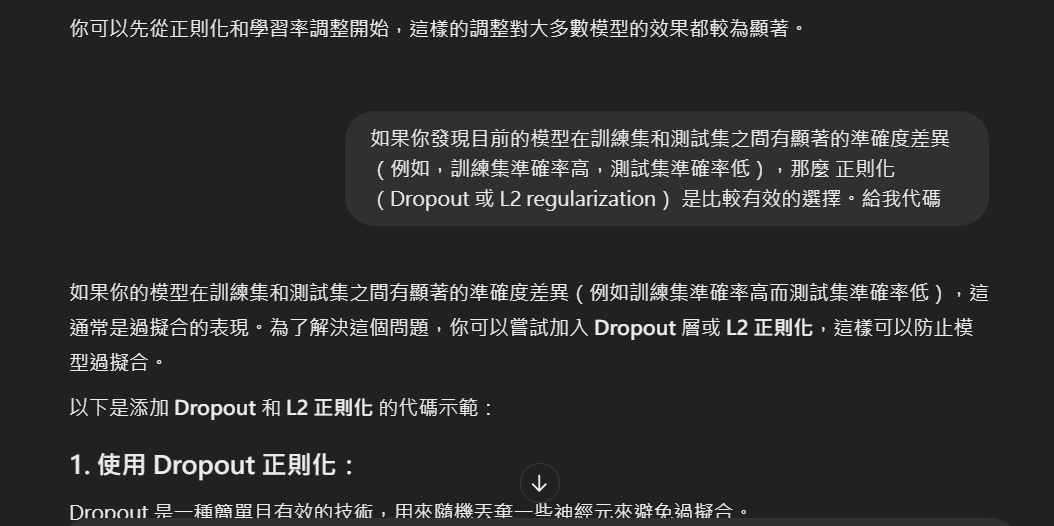
我發現調整後更低，於是維持第一版模型。

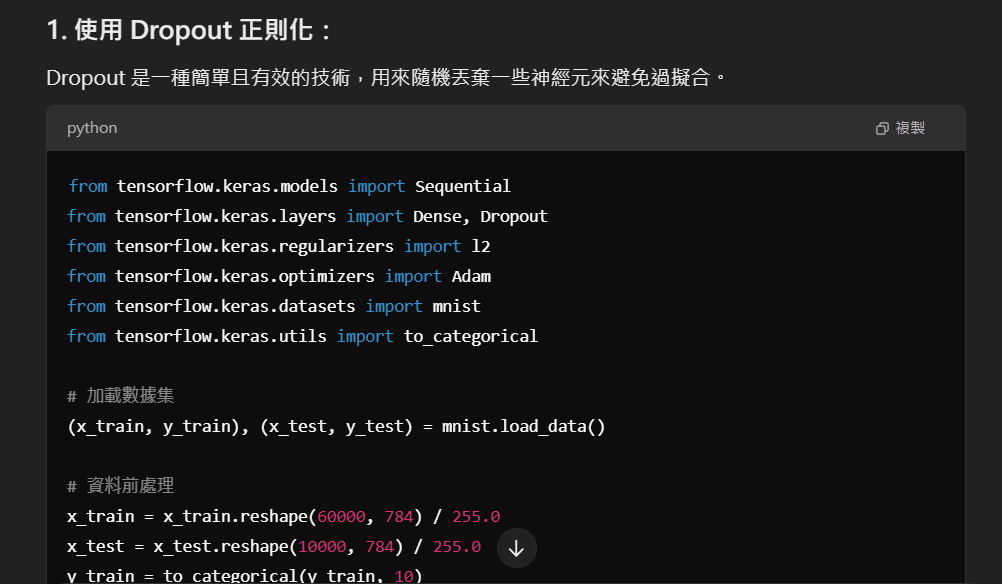
**截圖記錄:  
**

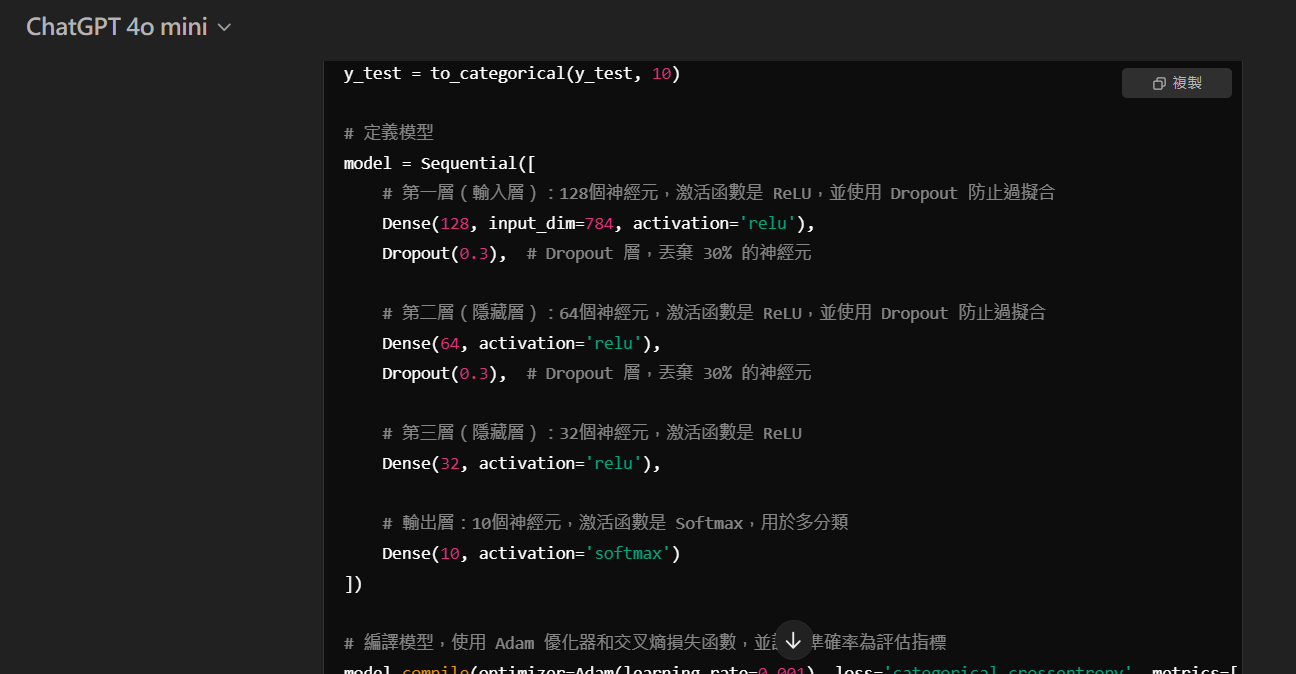
****

****

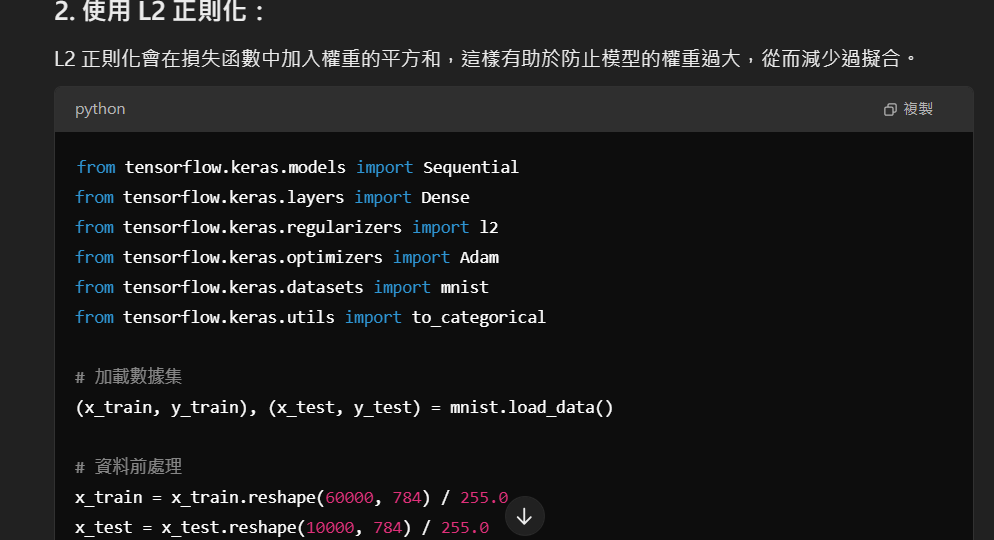
****

****

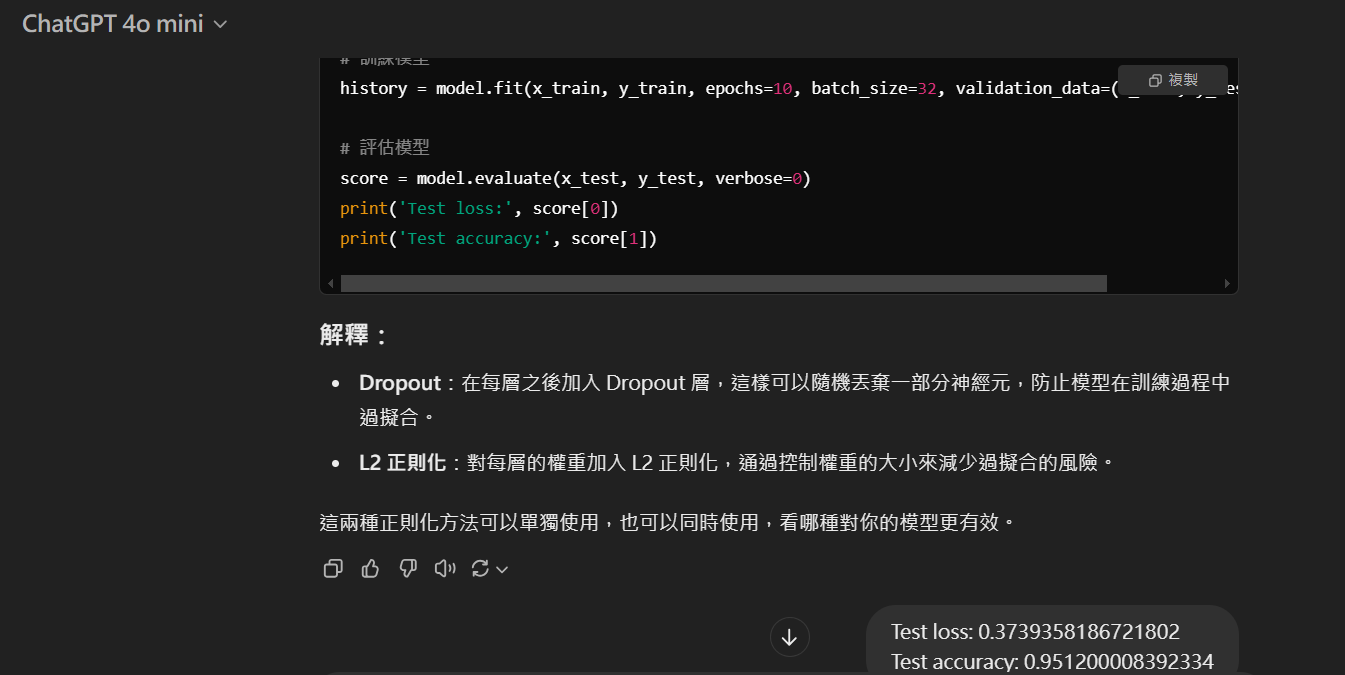
****

****

****

****

****

****

**文字版本紀錄:**

ChatGPT 說：

如果你的模型在測試集上已經達到約 97.89% 的準確率，這是一個相當不錯的結果，因此可以考慮進一步微調一些細節來提升模型的性能。根據現有的情況，以下幾個方法可能會比較有效：

增加層數或改變層的結構：

目前你的模型有 3 層隱藏層，你可以嘗試增加更多的隱藏層或調整每層的神經元數量。更多的層數可能有助於捕捉更複雜的模式，但需要小心避免過擬合。你可以嘗試在不過多增加計算負擔的情況下，增加 1-2 層，並觀察是否能提高測試準確率。

使用正則化（Dropout 或 L2 regularization）：

當模型達到較高準確率時，可能會遇到過擬合的問題。加入 Dropout 層可以幫助隨機丟棄一些神經元，從而減少過擬合。這可以讓模型學到更加穩定的特徵，並且能在測試集上表現得更好。

調整學習率：

雖然 Adam 是自適應學習率的優化器，但你可以嘗試手動設置一個學習率，例如 0.0001 或 0.005，並觀察對模型訓練過程的影響。有時候學習率過高會讓模型震盪或錯過最佳解，而過低則會收斂太慢。

資料增強（Data Augmentation）：

如果你覺得數據集有限，資料增強可能會幫助提升模型的泛化能力。這樣可以讓模型見識到更多的變化，並且能在測試集上獲得更高的準確率。你可以通過旋轉、平移、縮放等方式增加數據的多樣性。

哪一個最有效？

如果你發現目前的模型在訓練集和測試集之間有顯著的準確度差異（例如，訓練集準確率高，測試集準確率低），那麼 正則化（Dropout 或 L2 regularization） 是比較有效的選擇。

如果準確率已經相對穩定並接近極限，那麼 增加層數或神經元數量 可能會有效，但也要小心過擬合。

如果模型的訓練過程不穩定或準確率停滯，則可以嘗試 調整學習率，看看是否能夠加速收斂。

你可以先從正則化和學習率調整開始，這樣的調整對大多數模型的效果都較為顯著。

你說：

如果你發現目前的模型在訓練集和測試集之間有顯著的準確度差異（例如，訓練集準確率高，測試集準確率低），那麼 正則化（Dropout 或 L2 regularization） 是比較有效的選擇。給我代碼

ChatGPT 說：

如果你的模型在訓練集和測試集之間有顯著的準確度差異（例如訓練集準確率高而測試集準確率低），這通常是過擬合的表現。為了解決這個問題，你可以嘗試加入 Dropout 層或 L2 正則化，這樣可以防止模型過擬合。

以下是添加 Dropout 和 L2 正則化 的代碼示範：

1. 使用 Dropout 正則化：

Dropout 是一種簡單且有效的技術，用來隨機丟棄一些神經元來避免過擬合。

python

複製

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from tensorflow.keras.regularizers import l2

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to\_categorical

# 加載數據集

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

# 資料前處理

x\_train = x\_train.reshape(60000, 784) / 255.0

x\_test = x\_test.reshape(10000, 784) / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# 定義模型

model = Sequential([

# 第一層（輸入層）：128個神經元，激活函數是 ReLU，並使用 Dropout 防止過擬合

Dense(128, input\_dim=784, activation='relu'),

Dropout(0.3), # Dropout 層，丟棄 30% 的神經元

# 第二層（隱藏層）：64個神經元，激活函數是 ReLU，並使用 Dropout 防止過擬合

Dense(64, activation='relu'),

Dropout(0.3), # Dropout 層，丟棄 30% 的神經元

# 第三層（隱藏層）：32個神經元，激活函數是 ReLU

Dense(32, activation='relu'),

# 輸出層：10個神經元，激活函數是 Softmax，用於多分類

Dense(10, activation='softmax')

])

# 編譯模型，使用 Adam 優化器和交叉熵損失函數，並設置準確率為評估指標

model.compile(optimizer=Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

# 訓練模型

history = model.fit(x\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(x\_test, y\_test))

# 評估模型

score = model.evaluate(x\_test, y\_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

2. 使用 L2 正則化：

L2 正則化會在損失函數中加入權重的平方和，這樣有助於防止模型的權重過大，從而減少過擬合。

python

複製

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.regularizers import l2

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.datasets import mnist

from tensorflow.keras.utils import to\_categorical

# 加載數據集

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

# 資料前處理

x\_train = x\_train.reshape(60000, 784) / 255.0

x\_test = x\_test.reshape(10000, 784) / 255.0

y\_train = to\_categorical(y\_train, 10)

y\_test = to\_categorical(y\_test, 10)

# 定義模型

model = Sequential([

# 第一層（輸入層）：128個神經元，激活函數是 ReLU，並使用 L2 正則化

Dense(128, input\_dim=784, activation='relu', kernel\_regularizer=l2(0.01)),

# 第二層（隱藏層）：64個神經元，激活函數是 ReLU，並使用 L2 正則化

Dense(64, activation='relu', kernel\_regularizer=l2(0.01)),

# 第三層（隱藏層）：32個神經元，激活函數是 ReLU，並使用 L2 正則化

Dense(32, activation='relu', kernel\_regularizer=l2(0.01)),

# 輸出層：10個神經元，激活函數是 Softmax，用於多分類

Dense(10, activation='softmax')

])

# 編譯模型，使用 Adam 優化器和交叉熵損失函數，並設置準確率為評估指標

model.compile(optimizer=Adam(learning\_rate=0.001), loss='categorical\_crossentropy', metrics=['accuracy'])

# 訓練模型

history = model.fit(x\_train, y\_train, epochs=10, batch\_size=32, validation\_data=(x\_test, y\_test))

# 評估模型

score = model.evaluate(x\_test, y\_test, verbose=0)

print('Test loss:', score[0])

print('Test accuracy:', score[1])

解釋：

Dropout：在每層之後加入 Dropout 層，這樣可以隨機丟棄一部分神經元，防止模型在訓練過程中過擬合。

L2 正則化：對每層的權重加入 L2 正則化，通過控制權重的大小來減少過擬合的風險。

這兩種正則化方法可以單獨使用，也可以同時使用，看哪種對你的模型更有效。

你說：

Test loss: 0.3739358186721802

Test accuracy: 0.951200008392334