國立臺南大學

資訊工程學系

機器學習作業四

指導教授：李健興教授

資工四 S11059003 曾俊杰

西元 2024 年 10 月 12 日

目錄

圖目錄

1. 主程式

(1) DNN

如附件 S11059003_DNN.ipynb

(2) DCNN

如附件 S11059003_DCNN.ipynb

2. 執行過程

(1) 共通程式

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.utils import get_file
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.models import Sequential
from keras.utils import to_categorical

# 下載MNIST數據集
mnist_file = get_file('mnist.npz', origin='https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz')
mnist = np.load(mnist_file)

# 數據預處理
train_images, train_labels = mnist['x_train'], mnist['y_train']
test_images, test_labels = mnist['x_test'], mnist['y_test']

# Const
data_len = 28
data_wid = 28
input_size = data_len * data_wid
output_size = 10
initial = 5
enternal = 11
input_layer = int(pow(2,initial))
```

圖 1：共通程式圖

(2) DNN

```
# 數據預處理
train_images = train_images.reshape((-1, input_size)) / 255.0
test_images = test_images.reshape((-1, input_size)) / 255.0

# one-hot encoding
train_labels = to_categorical(train_labels, num_classes=output_size)
test_labels = to_categorical(test_labels, num_classes=output_size)

# 構建CNN模型

input_shape = (data_len, data_wid, 1)
model = models.Sequential()
model.add(Dense(input_layer, activation='relu', input_dim=input_size))
for i in range(initial+1, enternal):
    for j in range(3):
        model.add(Dense(int(pow(2,i)),activation='relu'))
model.add(Dense(output_size, activation='softmax'))

model.summary()
```

圖 2：DNN 預處理和模型

i. 數據預處理：將大小為 28×28 二維資料，展開為 784 一維資料，並正規化，除以 255 將 784 個數字都限制在[0, 1]。

ii. One-Hot encoding：將類別標籤轉換為二進制向量，使得神經網絡能夠進行分類。

iii. 構建 DNN 模型：寫出輸入層、隱藏層(Hidden layer)每個維度 3 層、輸出層。輸入和隱藏層的激勵函數設為 ReLU；輸出層的激勵函數則設為 Softmax，每層維度設定、參數統計(model.summary())如下圖。

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 32)	25,120
dense_1 (Dense)	(None, 64)	2,112
dense_2 (Dense)	(None, 64)	4,160
dense_3 (Dense)	(None, 64)	4,160
dense_4 (Dense)	(None, 128)	8,320
dense_5 (Dense)	(None, 128)	16,512
dense_6 (Dense)	(None, 128)	16,512
dense_7 (Dense)	(None, 256)	33,024
dense_8 (Dense)	(None, 256)	65,792
dense_9 (Dense)	(None, 256)	65,792
dense_10 (Dense)	(None, 512)	131,584
dense_11 (Dense)	(None, 512)	262,656
dense_12 (Dense)	(None, 512)	262,656
dense_13 (Dense)	(None, 1024)	525,312
dense_14 (Dense)	(None, 1024)	1,049,600
dense_15 (Dense)	(None, 1024)	1,049,600
dense_16 (Dense)	(None, 10)	10,250

Total params: 3,533,162 (13.48 MB)
Trainable params: 3,533,162 (13.48 MB)
Non-trainable params: 0 (0.00 B)

圖 3：DNN 模型 summary

```
# TODO 訓練模型
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
history = model.fit(train_images, train_labels, validation_data=(test_images, test_labels), epochs=20, batch_size=200)

# 創建一個具有兩個子圖的圖形
fig, axes = plt.subplots(1, 2, figsize=(15, 5))

# 第一個子圖 - 訓練的損失和準確率變化
axes[0].plot(history.history['accuracy'], label='Training Accuracy')
axes[0].plot(history.history['val_accuracy'], label='Validation Accuracy')
axes[0].set_xlabel('Epochs')
axes[0].set_title('Training Accuracy and Validation Accuracy')
axes[0].legend()

# 第二個子圖 - 驗證的損失和準確率變化
axes[1].plot(history.history['loss'], label='Training Loss')
axes[1].plot(history.history['val_loss'], label='Validation Loss')
axes[1].set_xlabel('Epochs')
axes[1].set_title('Training Loss and Validation Loss')
axes[1].legend()

# 設置兩個子圖之間的間距
plt.tight_layout()

# 顯示圖形
plt.show()
```

圖 4：DNN 訓練模型設定

i. 訓練模型：使用 Adam 優化器、分類交叉熵作為損失函數、準確率來評估模型的性能、迭代次數設定為 20、每次訓練處理 200 筆資料。

ii. 繪製圖形：繪製在每個迭代中準確率、損失函數的折線圖。備註：範例程式碼中，兩個子圖的 set_title(內容)寫反了。

(3) DCNN

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dropout
```

圖 5：DCNN 額外 module

```
filter_size = (3,3)
pooling_size = (2,2)
pooling_times = 2
flatten = int(np.sqrt(int(pow(2,enternal-1))))

# 數據預處理
train_images, test_images = train_images / 255.0, test_images / 255.0

# one-hot encoding
train_labels = to_categorical(train_labels, num_classes=output_size)
test_labels = to_categorical(test_labels, num_classes=output_size)
```

圖 6：DCNN 額外 const 和預處理

i. Const：包含 DNN 中的 const 和圖中額外的 const

ii. 數據預處理：將大小為 28×28 二維資料，不用展開，直接正規化，除以 255 將 784 個數字都限制在[0, 1]。

iii. One-Hot encoding：將類別標籤轉換為二進制向量，使得神經網絡能夠進行分類。

```
# repeat model
def layer(dim, size, input_dim, max_pool):
    global pooling_times
    if input_dim != None:
        model.add(Conv2D(dim, size, activation='relu', input_shape=input_dim))
    else:
        model.add(Conv2D(dim, size, activation='relu'))

    if max_pool != None and pooling_times > 0:
        pooling_times -= 1
        model.add(MaxPooling2D(max_pool))

# 構建DCNN模型
input_shape = (data_len, data_wid, 1)

model = models.Sequential()

layer(input_layer, filter_size, input_shape, pooling_size)
for i in range(initial+1, enternal):
    for j in range(1):
        layer(int(pow(2,i)), filter_size, None, pooling_size)

model.add(Flatten())
model.add(Dense(flatten, activation='relu'))
model.add(Dense(output_size, activation='softmax'))

model.summary()
```

圖 7：DCNN 模型

i. Repeat model：layer(輸入維度, 過濾器大小, 輸入維度大小, 最大池化大小)，如果存在輸入維度大小，則是輸入層，如果存在最大池化大小，該層需要做最大池化，最大池化次數為 2 次，也就是輸入層和第一個隱藏層做最大池化。

ii. 建構 DCNN 模型：寫出輸入層、隱藏層(Hidden layer)每個維度 1 層、輸出層。輸入和隱藏層的激勵函數設為 ReLU；輸出層的激勵函數則設為 Softmax，每層維度設定、參數統計(model.summary())如下圖。

Model: "sequential_4"

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_8 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_16 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_9 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_17 (Conv2D)	(None, 3, 3, 128)	73,856
conv2d_18 (Conv2D)	(None, 1, 1, 256)	295,168
flatten_4 (Flatten)	(None, 256)	0
dense_8 (Dense)	(None, 16)	4,112
dense_9 (Dense)	(None, 10)	170

Total params: 392,122 (1.50 MB)
Trainable params: 392,122 (1.50 MB)
Non-trainable params: 0 (0.00 B)

圖 8：DCNN 模型 summary

```
history = model.fit(train_images, train_labels, validation_data=(test_images, test_labels), epochs=20, batch_size=512)
```

圖 9：DCNN 訓練模型設定

i. DCNN 訓練模型設定與 DNN 相同，只有 batch_size 改成 512，每次訓練處理 512 筆資料。

3. 執行結果

(1) DNN

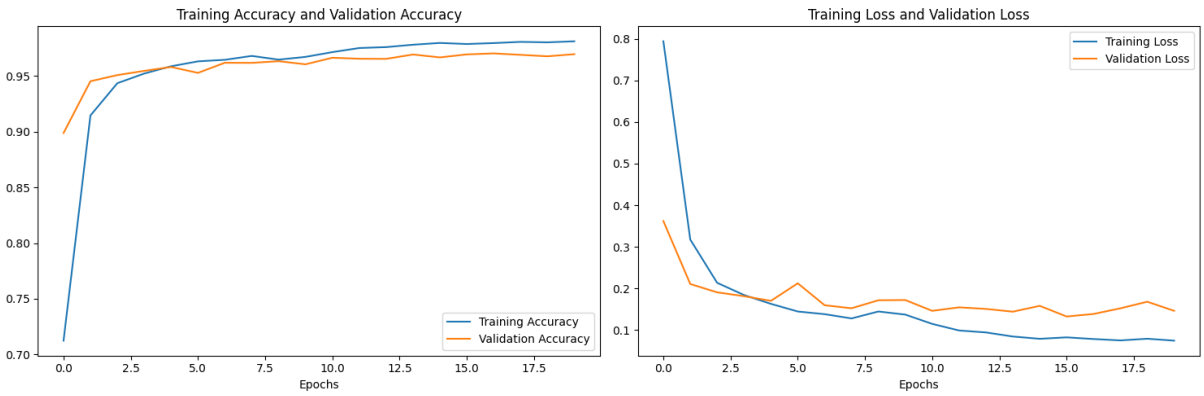

圖 10：DNN 每個迭代中準確率、損失函數折線圖

訓練準確率和測試(驗證)準確率趨勢相符，非常貼近，是不錯的訓練結果

i. 第一個 epoch 的訓練準確率為 0.5137，損失為 1.2652；最終訓練準確率為 0.9821，損失為 0.0716。

ii. 第一個 epoch 的測試準確率為 0.8989，損失為 0.3621；最終測試準確率為 0.9697，損失為 0.1463

iii. 全部 epoch 用時總和：1409 秒，平均 1409÷20 = 70.45 秒/epoch。

(2) DCNN

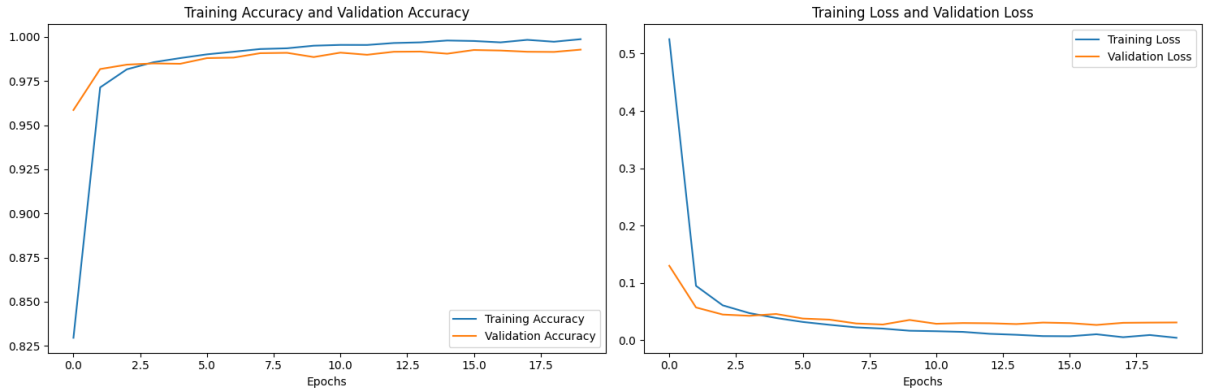

圖 11：DCNN 每個迭代中準確率、損失函數的折線圖

訓練準確率和測試(驗證)準確率趨勢相符，非常貼近，是不錯的訓練結果

i. 第一個 epoch 的訓練準確率為 0.6442，損失為 1.0551；最終訓練準確率為 0.9985，損失為 0.0048。

ii. 第一個 epoch 的測試準確率為 0.9586，損失為 0.1296；最終測試準確率為 0.9928，損失為 0.0307

iii. 全部 epoch 用時總和：2436 秒，平均 2436÷20 = 121.8 秒/epoch。

參考文獻

[1] ML Lecture 8-1: “Hello world” of deep learning，李宏毅教授，https://www.youtube.com/watch?v=Lx3l4lOrquw

[2] ML Lecture 8-3: Keras Demo，李宏毅教授，https://www.youtube.com/watch?v=L8nuuZNpWw8

[3] ML Lecture 9-2: Keras Demo 2，李宏毅教授，https://www.youtube.com/watch?v=Ky1ku1miDow&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=17

[4] ChatGPT，OpenAI，ChatGPT