Lab 2 結報

姓名：仇健安　　學號：111511239

請敘述在課堂上實作之神經網路的架構

我在 Lab 2 中設計並比較了三種神經網路架構，分別為：

1. 基本題架構（無 shuffle，無 MaxPooling、Dropout、BatchNorm）
2. 半加分題架構（有 shuffle，無 MaxPooling、Dropout、BatchNorm）
3. 完整加分題架構（有 shuffle，並在每層 CNN 後加入 MaxPooling、BatchNormalization、Dropout）

【資料前處理】

- 資料來源：4 位使用者於 6 個位置的 CSI 訊號，共有 14400 筆訓練資料與 14400 筆測試資料。
- 讀取方式：透過 scipy.io.loadmat() 讀取 .mat 格式資料，並整理成形狀為 (14400, 56, 4) 的張量，其中 56 為子載波數，4 是發射與接收天線對數。
- 標籤處理：利用 OneHotEncoder() 將位置標籤轉為 one-hot 向量。
- shuffle（加分題限定）：將訓練資料隨機打亂，避免同類資料連續出現導致過度擬合。

【基本題神經網路架構】

Input: (56, 4)
→ Conv1D(32, kernel=3, activation=leaky_relu, padding='same')
→ Conv1D(64, kernel=3, activation=leaky_relu, padding='same')
→ Conv1D(128, kernel=3, activation=leaky_relu, padding='same')
→ Conv1D(128, kernel=3, activation=leaky_relu, padding='same')
→ Flatten (變成 56×128 = 7168 維向量)
→ Dense(256, activation=leaky_relu)
→ Dense(64, activation=leaky_relu)
→ Dense(6, activation=softmax)

- 總參數量很大，主因在於沒有縮減參數直接 Flatten 給 Dense 層。
- 沒有使用任何 Nomalization 機制，比較不好 train。
- 沒有隨機打亂資料，所以 validation acc 會很糟糕。

Model: "基本題"

Layer (type)	Output Shape	Param #
input_layer_7 (InputLayer)	(None, 56, 4)	0
conv1d_20 (Conv1D)	(None, 56, 32)	416
conv1d_21 (Conv1D)	(None, 56, 64)	6,208
conv1d_22 (Conv1D)	(None, 56, 128)	24,704
conv1d_23 (Conv1D)	(None, 56, 128)	49,280
flatten_5 (Flatten)	(None, 7168)	0
dense_15 (Dense)	(None, 256)	1,835,264
dense_16 (Dense)	(None, 64)	16,448
dense_17 (Dense)	(None, 6)	390

Total params: 1,932,710 (7.37 MB)
Trainable params: 1,932,710 (7.37 MB)
Non-trainable params: 0 (0.00 B)

【半加分題架構】

- 模型與基本題完全相同，但在訓練前打亂訓練資料順序，能改善模型偏見與提升泛化能力。

```
permutation = np.random.permutation(train_dataset.shape[0])
train_dataset = train_dataset[permutation]
training_label = training_label[permutation]
```

Model: "半加分題"

Layer (type)	Output Shape	Param #
input_layer_8 (InputLayer)	(None, 56, 4)	0
conv1d_24 (Conv1D)	(None, 56, 32)	416
conv1d_25 (Conv1D)	(None, 56, 64)	6,208
conv1d_26 (Conv1D)	(None, 56, 128)	24,704
conv1d_27 (Conv1D)	(None, 56, 128)	49,280
flatten_6 (Flatten)	(None, 7168)	0
dense_18 (Dense)	(None, 256)	1,835,264
dense_19 (Dense)	(None, 64)	16,448
dense_20 (Dense)	(None, 6)	390

Total params: 1,932,710 (7.37 MB)

Trainable params: 1,932,710 (7.37 MB)

Non-trainable params: 0 (0.00 B)

【完整加分題架構】

Input: (56, 4)
→ Conv1D(32, 3, activation=leaky_relu)
→ MaxPooling1D(2)
→ BatchNormalization()
→ Dropout(0.3)

→ Conv1D(64, 3, activation=leaky_relu)
→ MaxPooling1D(2)
→ BatchNormalization()
→ Dropout(0.3)

→ Conv1D(128, 3, activation=leaky_relu)
→ MaxPooling1D(2)
→ BatchNormalization()
→ Dropout(0.3)

→ Conv1D(128, 3, activation=leaky_relu)
→ MaxPooling1D(2)
→ BatchNormalization()
→ Dropout(0.3)

→ Flatten (維度大幅下降)
→ Dense(256, activation=leaky_relu)
→ Dense(64, activation=leaky_relu)
→ Dense(6, activation=softmax)

- 每層都加上 MaxPooling、BN、Dropout。
- 訓練穩定， 且參數量因 MaxPooling 降維而顯著減少。

Model: "完整加分題"

Layer (type)	Output Shape	Param #
input_layer_9 (InputLayer)	(None, 56, 4)	0
conv1d_28 (Conv1D)	(None, 56, 32)	416
max_pooling1d_8 (MaxPooling1D)	(None, 28, 32)	0
batch_normalization_8 (BatchNormalization)	(None, 28, 32)	128
spatial_dropout1d_8 (SpatialDropout1D)	(None, 28, 32)	0
conv1d_29 (Conv1D)	(None, 28, 64)	6,208
max_pooling1d_9 (MaxPooling1D)	(None, 14, 64)	0
batch_normalization_9 (BatchNormalization)	(None, 14, 64)	256
spatial_dropout1d_9 (SpatialDropout1D)	(None, 14, 64)	0
conv1d_30 (Conv1D)	(None, 14, 128)	24,704
max_pooling1d_10 (MaxPooling1D)	(None, 7, 128)	0
batch_normalization_10 (BatchNormalization)	(None, 7, 128)	512
spatial_dropout1d_10 (SpatialDropout1D)	(None, 7, 128)	0
conv1d_31 (Conv1D)	(None, 7, 128)	49,280
max_pooling1d_11 (MaxPooling1D)	(None, 4, 128)	0
batch_normalization_11 (BatchNormalization)	(None, 4, 128)	512
spatial_dropout1d_11 (SpatialDropout1D)	(None, 4, 128)	0
flatten_7 (Flatten)	(None, 512)	0
dense_21 (Dense)	(None, 256)	131,328
dense_22 (Dense)	(None, 64)	16,448
dense_23 (Dense)	(None, 6)	390

Total params: 230,182 (899.15 KB)

Trainable params: 229,478 (896.40 KB)

Non-trainable params: 704 (2.75 KB)

請敘述 Maxpooling、Dropout 與 Batch Normalization 的作用

【MaxPooling】

- 功能：將 feature map 中的特徵區域取最大值，降低空間維度，保留主要特徵。
- 範例：原輸出為 (56, 128)，經過 MaxPooling1D(pool_size=2) → (28, 128)
- 優點：
  - 降低參數數量與計算成本。
  - 增加模型的平移不變性。
  - 有助於防止過擬合。

【Dropout】

- 功能：在訓練時隨機將某些神經元輸出設為 0。
- 範例：Dropout(0.3) 表示每次訓練隨機屏蔽 30% 的神經元。
- 優點：
  - 防止模型過度依賴某些節點。
  - 提升模型泛化能力。

【Batch Normalization】

- 功能：對每一層的輸出做標準化處理，使其均值為 0、標準差為 1。
- 計算：
  對 mini-batch 的每個 feature：
  $\hat{x} = (x - \mu) / \text{sqrt}(\sigma^2 + \varepsilon)$
- 優點：
  - 加快收斂速度。
  - 減少初始化依賴。
  - 緩解梯度消失問題。

心得

本次 Lab 實作讓我了解 1D-CNN 如何應用於無線訊號處理與室內定位，並透過對比不同架構的效果，實際體會資料前處理（shuffle）與正規化技巧（Pooling、Dropout、BN）對模型訓練的關鍵影響。我也更加熟悉 TensorFlow 的建模流程與訓練視覺化方式，對未來實作 AI 無線應用奠定基礎。

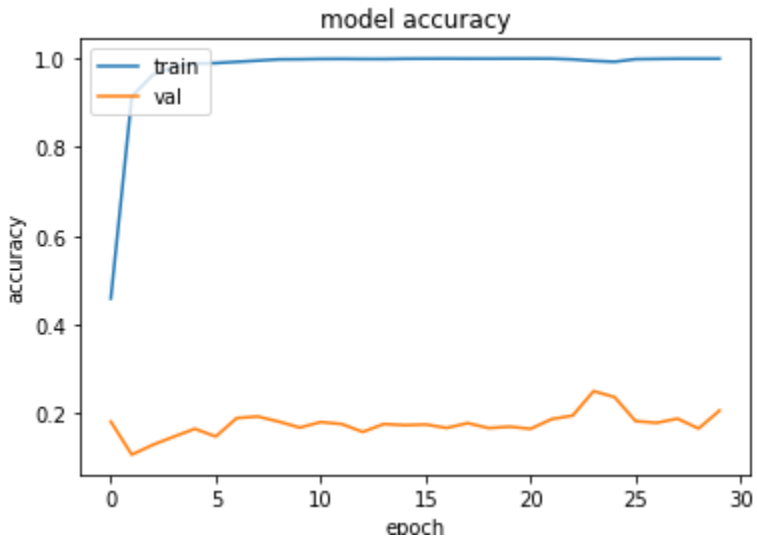
訓練結果

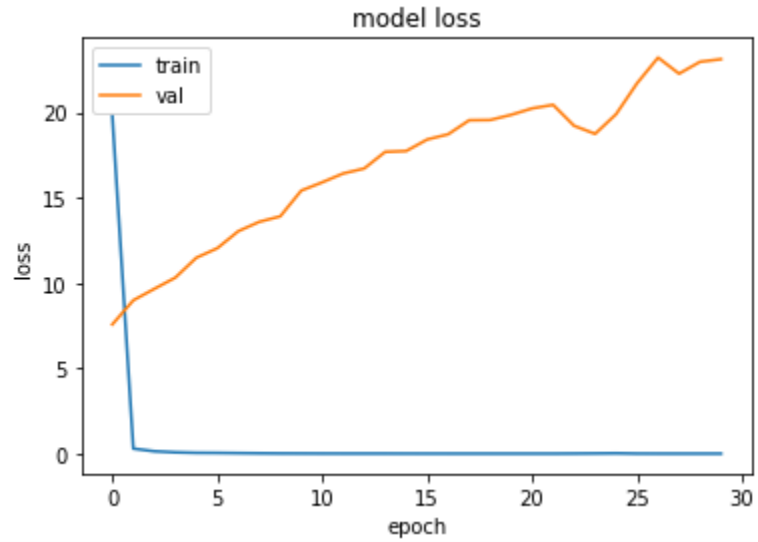
【基本題結果】

測試資料籍結果

```
57/57 [==============================] - 1s 11ms/step - loss: 8.2760 - accuracy: 0.6837

Confusion Matrix:
[[1684  125  375  216    0    0]
 [  69 2013  236   81    0    1]
 [ 101  381 1733   34  151    0]
 [   5  397    0 1448  166  384]
 [ 521   20   23   60 1776    0]
 [   0  684    0  524    0 1192]]
```



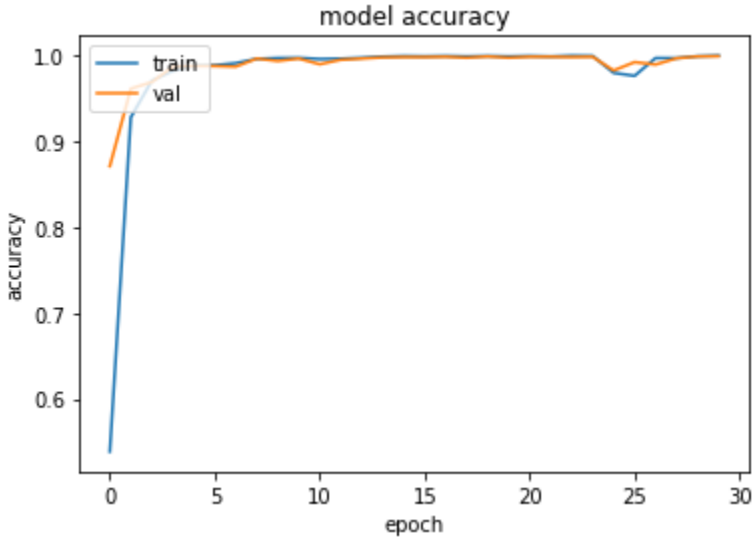


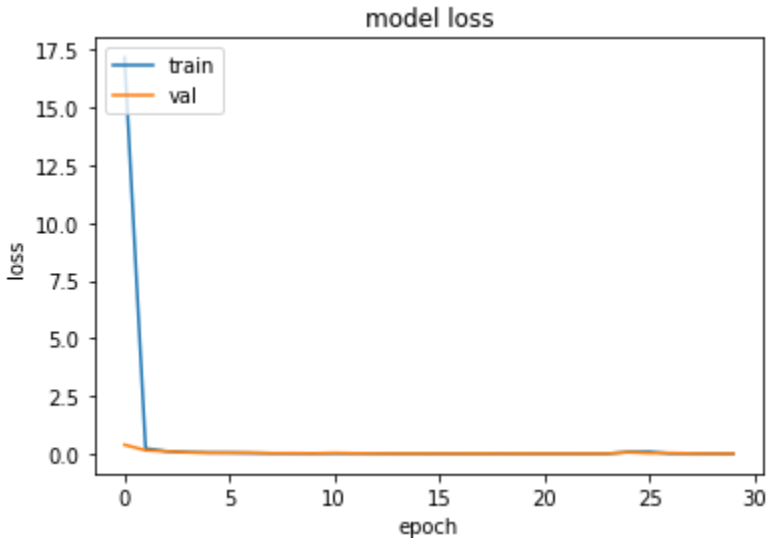
【半加分題結果】

測試資料籍結果

```
57/57 [==============================] - 1s 10ms/step - loss: 1.8574 - accuracy: 0.8086

Confusion Matrix:
[[2233   23  118   21    0    5]
 [  69 2100   15  178   34    4]
 [  66   20 1755   14  545    0]
 [   2  338    0 1945  115    0]
 [ 287    0   14  174 1925    0]
 [   0  640    0   74    0 1686]]
```





【完整加分題結果】

測試資料籍結果

```
57/57 [==============================] - 0s 5ms/step - loss: 1.3182 - accuracy: 0.8395

Confusion Matrix:
[[2159    0  239    0    0    2]
 [  36 2133    1   64   71   95]
 [ 113   15 1997   40  235    0]
 [   3  331    0 1773  250   43]
 [ 107    1    5   15 2272    0]
 [  14  580    0   51    0 1755]]
```