

第七章

深入探討Keras

蘇櫟

探討電腦視覺、時間序列預測、自然語言處理、 生成式深度學習

@reallygreatsite

章節

7-1 Keras工作流程

7-2 建構Keras模型不同方法

7-3 使用內建的訓練與評估迴圈

7-4 設計自己的訓練與評估迴圈

7-1 Keras工作流程

逐步提升複雜度

簡單-->簡單方法

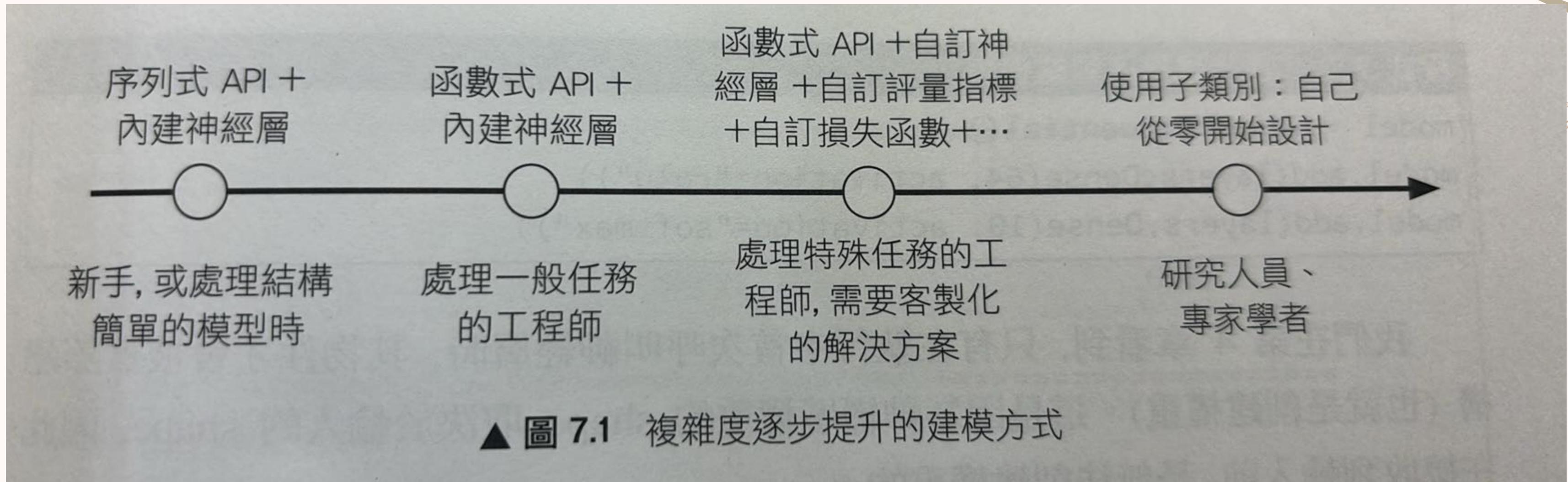
複雜-->複雜方法

7-2 建構Keras模型 的不同方式

1.序列式模型

2.函數式API

3.繼承Model類別



7-2-1 序列式模型

@reallygreatsite

7-2-2 函數式API

需要處理多輸出或多輸入，或是中間有很多分支的非線性拓樸。

範例一

範例二 多數入多輸出模型

評估消費者意見單優先性

輸入

1. 意見單標題
2. 意見單內容
3. 消費者勾選標籤

輸出

1. 優先性分數
2. 處理意見單部門

訓練多輸入多輸出模型

以輸入資料和輸出資料的list呼叫fit()

檢視神經層的連接方式

模型視覺化
特徵萃取

7-2-3 繼承Model類別

1. 在`__init__`中，定義模型會用到的神經層
2. 在`call()`中，定義個神經層的正向傳播
3. 實例化定義好的子類別

缺點

負責更多模型邏輯，淺在錯誤機會變高

7-2-4 混合搭配不同的 設計模式

7-2-5 根據任務挑選 適當工具

函數式API:易用性與彈性

使用Layer子類別解決特殊功能

7-3 使用內建的訓練 與評估模型

逐步提升複雜度

compile(), fit(), evaluate(), predict()

調整面相:

- 1.自定義的評量指標
- 2.將回乎物件傳遞給fit()

7-3-1 設計自己的評量指標

逐步提升複雜度

compile(), fit(), evaluate(), predict()

調整面相:

- 1.自定義的評量指標
- 2.將回乎物件傳遞給fit()

7-3-2 使用回呼模組

1. 模型檢查點

2. 早期停止

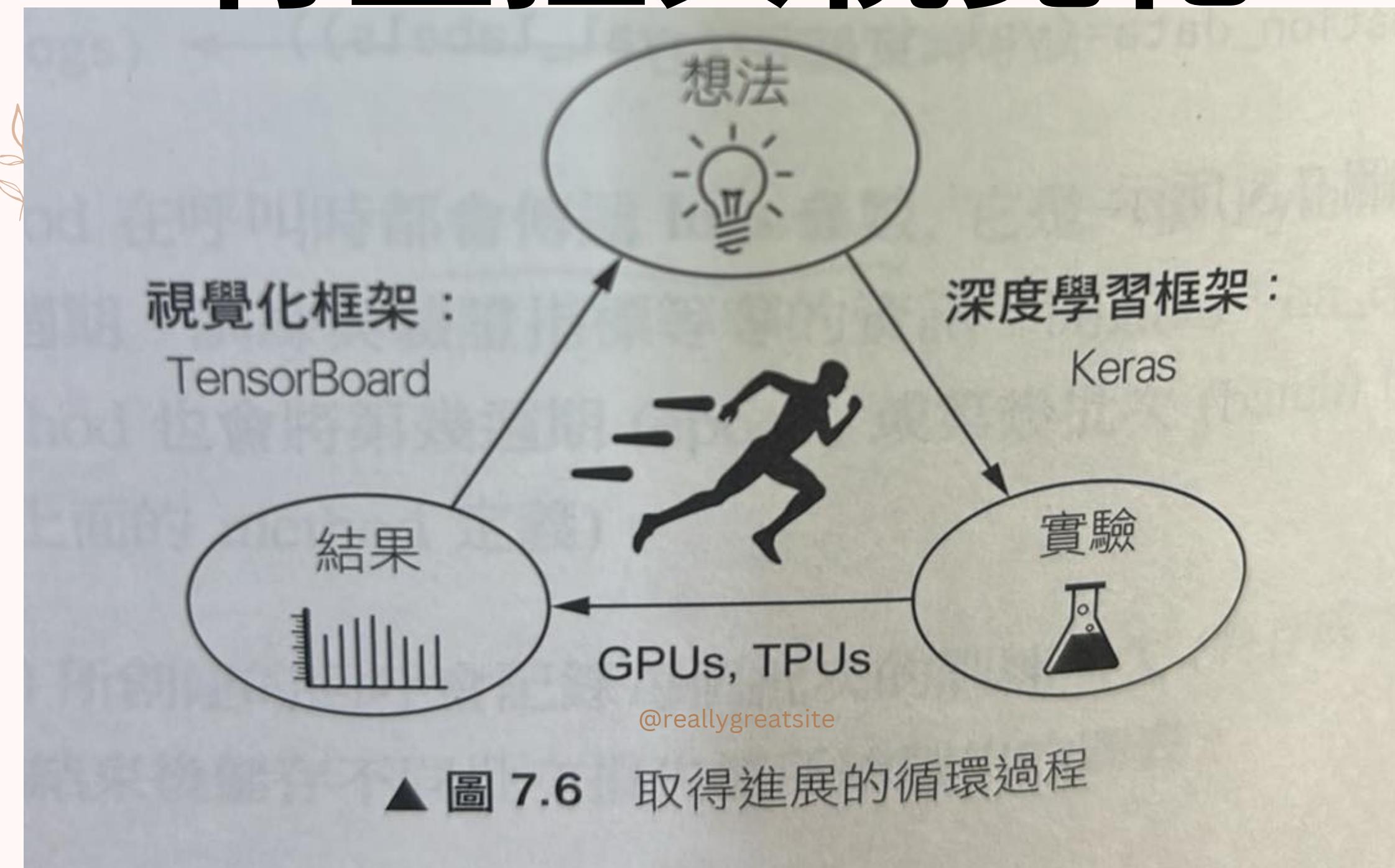
3. 動態調整參數

4. 紀錄訓練和驗證指標值

7-3-3 設計自己的回呼

1. `on_epoch_begin(epoch,logs)`: 訓練週期開始呼叫
2. `on_epoch_end(epoch,logs)`: 訓練週期結束呼叫
3. `on_batch_begin(batch,logs)`: 批次資料前呼叫
4. `on_batch_end(batch,logs)`: 批次資料後呼叫
5. `on_train_begin(logs)`: 訓練開始時呼叫
6. `on_epoch_end(logs)`: 訓練結束時呼叫

7-3-4 利用TensorBoard進行監控與視覺化



TensorBoard

- 
- 
1. 訓練期間，視覺化方式監控損失和評量指標
 2. 模型架構視覺化
 3. 結果和梯度變化以視覺化直方圖呈現
 4. 以3D方式探索嵌入向量

7-4 設計自己的訓練 及評估迴圈

1. 在梯度磁帶區塊中執行正向傳播
2. 取的損失相對於模型權重的梯度
3. 更新模型權重，降低當前批次的損失值

7-4-1 訓練VS推論

呼叫dropout(input,training=True)會拋棄參數
(訓練階段)

呼叫dropout(input,training=False)甚麼都不會
做(推論階段)

兩種權重：

1. 可訓練權重：反向傳播時進行更新降低損失值

2. 非訓練權重：正向傳播時更新

考慮了以上兩個細節後，監督式學習的訓練程式就變成：

```
def train_step(inputs, targets):
    with tf.GradientTape() as tape:
        predictions = model(inputs, training=True)
        loss = loss_fn(targets, predictions)
        gradients = tape.gradients(loss, model.trainable_weights)
    optimizer.apply_gradients(zip(model.trainable_weights, gradients))
```

training 參數要設為 True
處理可訓練權重

7-4-2 評量指標低階用法

```
>>> metric = keras.metrics.SparseCategoricalAccuracy() ← 創建一個評量  
>>> targets = [0, 1, 2]  
>>> predictions = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]  
>>> metric.update_state(targets, predictions) ← 更新評量指標的內部狀態  
>>> current_result = metric.result() ← 取得當前的指標值  
>>> print(f"result: {current_result:.2f}")  
result: 1.00
```

有時，我們想要追蹤某個純量（例如訓練損失）的平均值，這時就可以使用 `keras.metrics.Mean()` 物件：

```
>>> values = [0, 1, 2, 3, 4]  
>>> mean_tracker = keras.metrics.Mean()  
>>> for value in values:  
>>>     mean_tracker.update_state(value)  
>>> print(f"Mean of values: {mean_tracker.result():.2f}")  
Mean of values: 2.00
```

7-4-3 完整訓練及 評估迴圈

@reallygreatsite

7-4-4 利用 tf.function來加速

@reallygreatsite

7-4-5 搭配fit()和自 定義的訓練迴圈

改寫Model類別的train_step()方法

*Thank
You*

@reallygreatsite