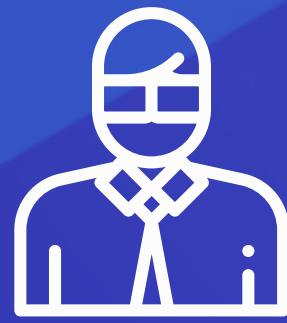




Day 69

Keras Module API

Keras Module API 的介紹與應用



陳宇春

出題教練



知識地圖 深度學習簡介

深度學習體驗 - 啟動函數與正規化

深度神經網路

Supervised Learning Deep Neural Network (DNN)

簡介 Introduction

套件介紹 Tools: Keras

組成概念 Concept

訓練技巧 Training Skill

應用案例 Application

卷積神經網路

Convolutional Neural Network (CNN)

簡介 introduction

套件練習 Practice with Keras

訓練技巧 Training Skill

電腦視覺 Computer Vision

深度學習套件介紹

Tools of DNN: Keras

Keras簡介與安裝

Keras 內建資料集下載

如何用 Keras 搭建類神經網路

本日知識點目標

- 了解 Keras Module API
- 了解 Keras Module API 與其應用的場景

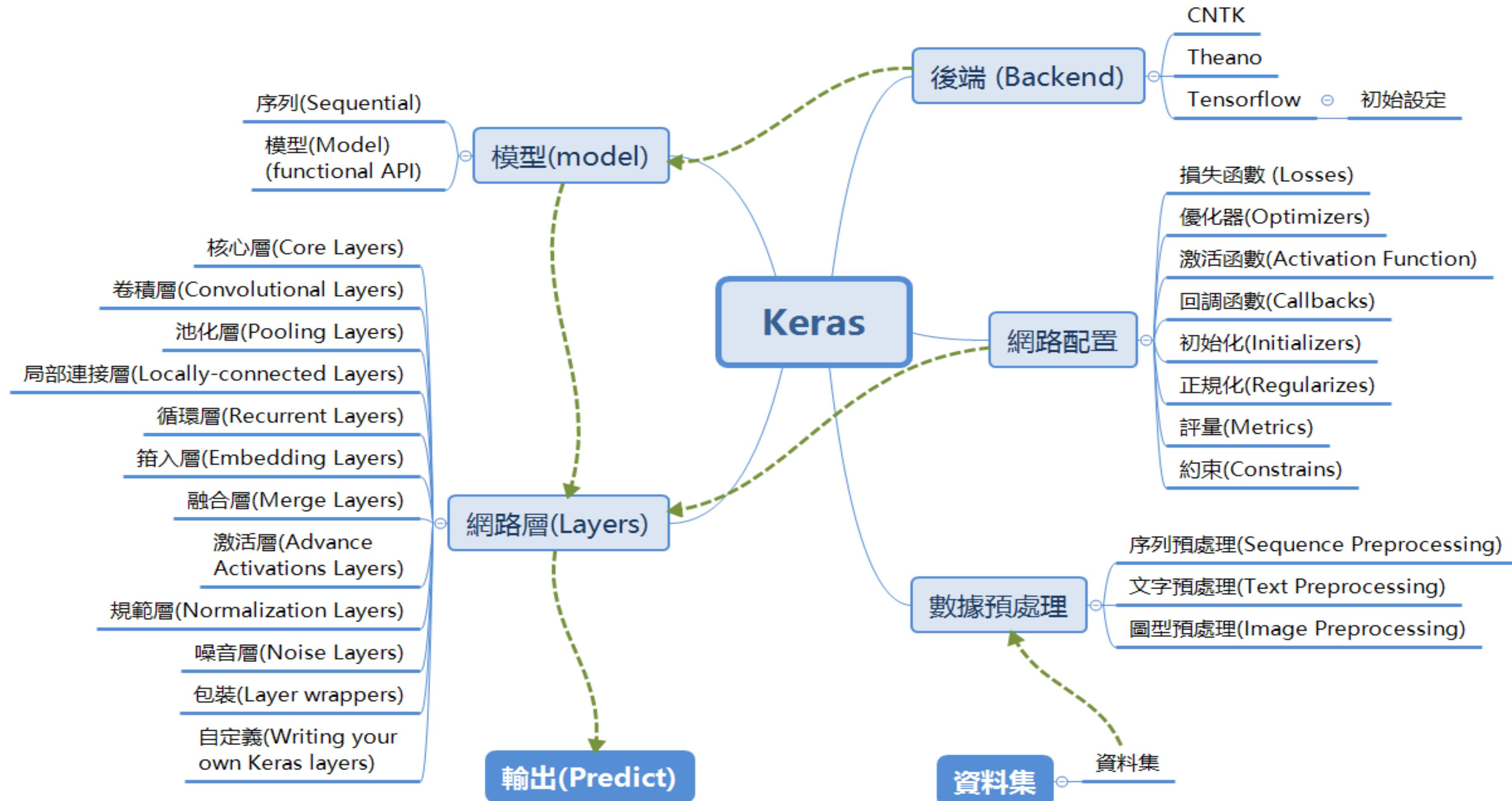
函數式API

- 用戶定義多輸出模型、非循環有向模型或具有共享層的模型等複雜模型的途徑
- 定義複雜模型（如多輸出模型、有向無環圖，或具有共享層的模型）的方法。
- 所有的模型都可調用，就像網絡層一樣
 - 利用函數式API，可以輕易地重用訓練好的模型：可以將任何模型看作是一個層，然後通過傳遞一個張量來調用它。注意，在調用模型時，您不僅重用模型的結構，還重用了它的權重。

範例

```
x = Input(shape=(784,))  
  
y = model(x)
```

Keras框架回顧



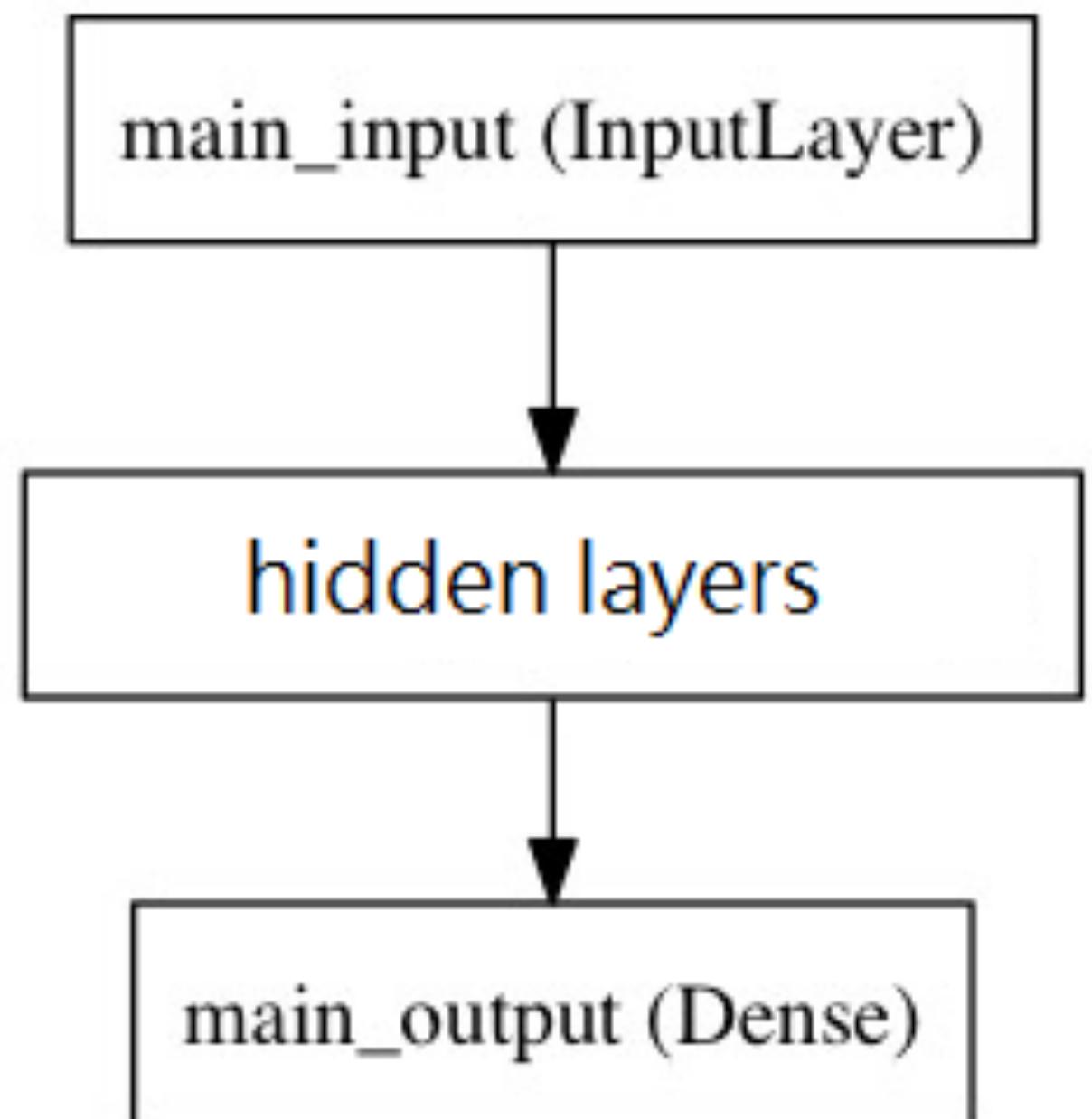
函數式API 與 順序模型

- 模型需要多於一個的輸出，那麼你總應該選擇函數式模型。
 - 函數式模型是最廣泛的一類模型，序貫模型（Sequential）只是它的一種特殊情況。
- 延伸說明
 - 層對象接受張量為參數，返回一個張量。
 - 輸入是張量，輸出也是張量的一個框架就是一個模型，通過Model定義。
 - 這樣的模型可以被像Keras的Sequential一樣被訓練。

如何配置

- 使用函數式模型的一個典型場景是搭建多輸入、多輸出的模型
- 使用如下：

```
from keras.layers import Input  
from keras.models import Model  
main_input = Input(shape=(100,), dtype='int32', name='main_input')
```



應用說明

- 利用函數式 API，可以輕易地重用訓練好的模型：可以將任何模型看作是一個層，然後通過傳遞一個張量來調用它。注意，在調用模型時，您不僅重用模型的結構，還重用了它的權重。
- 具有多個輸入和輸出的模型。函數式 API 使處理大量交織的數據流變得容易。
- 試圖預測 Twitter 上的一條新聞標題有多少轉發和點贊數

應用說明(II)



- 模型的主要輸入將是新聞標題本身，即一系列詞語。
- 但是為了增添趣味，我們的模型還添加了其他的輔助輸入來接收額外的數據，例如新聞標題的發布的時間等。
- 該模型也將通過兩個損失函數進行監督學習。較早地在模型中使用主損失函數，是深度學習模型的一個良好正則方法。

前述流程 / python 程式 對照

```
from keras.layers import Input, Embedding, LSTM, Dense
from keras.models import Model

# 主要輸入接收新聞標題本身，即一個整數序列（每個整數編碼一個詞）。
# 這些整數在1 到10,000 之間（10,000 個詞的詞彙表），且序列長度為100 個詞
# 宣告一個 NAME 去定義Input
main_input = Input(shape=(100,), dtype='int32', name='main_input')

# Embedding 層將輸入序列編碼為一個稠密向量的序列，
# 每個向量維度為 512。
x = Embedding(output_dim=512, input_dim=10000, input_length=100)(main_input)

# LSTM 層把向量序列轉換成單個向量，
# 它包含整個序列的上下文信息
lstm_out = LSTM(32)(x)
```

前述流程 / python 程式 對照

```
#插入輔助損失，使得即使在模型主損失很高的情況下，LSTM 層和Embedding 層都能被平穩地訓練
news_output = Dense(1, activation='sigmoid', name='news_out')(lstm_out)

#輔助輸入數據與LSTM 層的輸出連接起來，輸入到模型
import keras
news_input = Input(shape=(5,), name='news_in')
x = keras.layers.concatenate([lstm_out, news_input])

# 堆疊多個全連接網路層
x = Dense(64, activation='relu')(x)
x = Dense(64, activation='relu')(x)
#作業解答：新增兩層
x = Dense(64, activation='relu')(x)
x = Dense(64, activation='relu')(x)

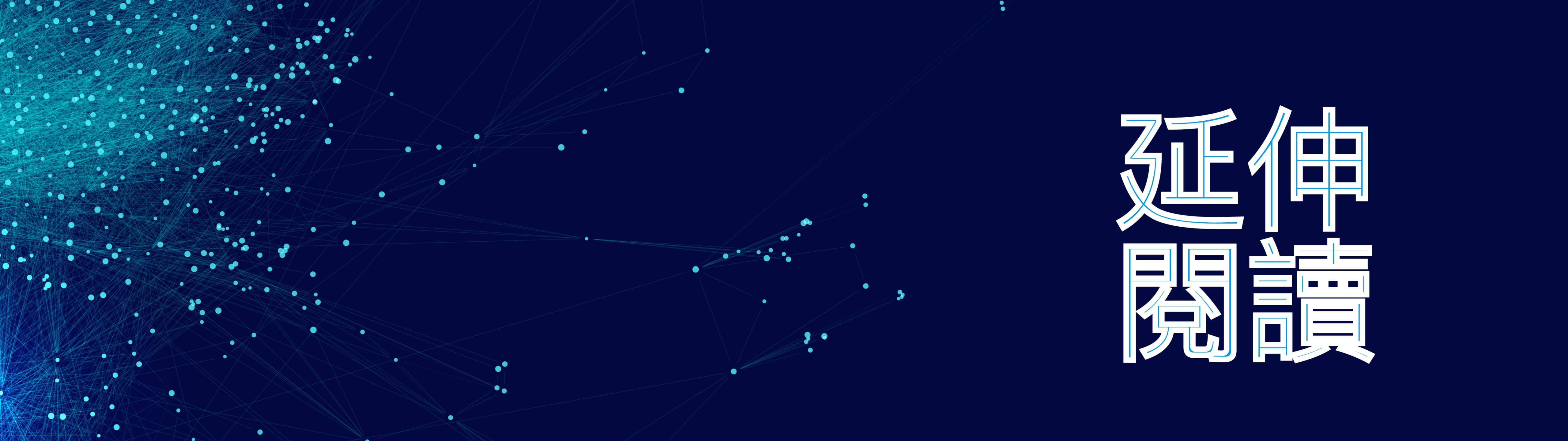
# 最後添加主要的邏輯回歸層
main_output = Dense(1, activation='sigmoid', name='main_output')(x)

# 宣告 MODEL API，分別採用自行定義的 Input/Output Layer
model = Model(inputs=[main_input, auxiliary_input], outputs=[main_output, auxiliary_output])

model.compile(optimizer='rmsprop',
              loss={'main_output': 'binary_crossentropy', 'aux_output': 'binary_crossentropy'},
              loss_weights={'main_output': 1., 'aux_output': 0.2})
```

重要知識點複習：共享網路層

- 函數式API的另一個用途是使用共享網絡層的模型。
- 我們來看看共享層。
 - 來考慮推特推文數據集。我們想要建立一個模型來分辨兩條推文是否來自同一個人（例如，通過推文的相似性來對用戶進行比較）。
 - 實現這個目標的一種方法是建立一個模型，將兩條推文編碼成兩個向量，連接向量，然後添加邏輯回歸層；這將輸出兩條推文來自同一作者的概率。模型將接收一對對正負表示的推特數據。
 - 由於這個問題是對稱的，編碼第一條推文的機制應該被完全重用來編碼第二條推文（權重及其他全部）。



延伸 閱讀

除了每日知識點的基礎之外，推薦的延伸閱讀能補足學員們對該知識點的了解程度，建議您解完每日題目後，若有
多餘時間，可再補充延伸閱讀文章內容。

推薦延伸閱讀

Getting started with the Keras functional API

層「節點」的概念

- 每當你在某個輸入上調用一個層時，都將創建一個新的張量（層的輸出），並且為該層添加一個「節點」，將輸入張量連接到輸出張量。當多次調用同一個圖層時，該圖層將擁有多個節點索引(0, 1, 2...)。
- 但是如果一個層與多個輸入連接呢？
 - 只要一個層僅僅連接到一個輸入，就不會有困惑，.output會返回層的唯一輸出

First example: a densely-connected network

The `Sequential` model is probably a better choice to implement such a network, but it helps to start with something really simple.

- A layer instance is callable (on a tensor), and it returns a tensor
- Input tensor(s) and output tensor(s) can then be used to define a `Model`
- Such a model can be trained just like Keras `Sequential` models.

```
from keras.layers import Input, Dense
from keras.models import Model

# This returns a tensor
inputs = Input(shape=(784,))

# a layer instance is callable on a tensor, and returns a tensor
output_1 = Dense(64, activation='relu')(inputs)
output_2 = Dense(64, activation='relu')(output_1)
predictions = Dense(10, activation='softmax')(output_2)

# This creates a model that includes
# the Input layer and three Dense layers
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels) # starts training
```

推薦延伸閱讀

全連接網路

- Sequential 模型可能是實現這種網絡的一個更好選
 - 擇網路層的實例是可調用的，它以張量為參數，並且返回一個張量
 - 入和輸出均為張量，它們都可以用來定義一個模型（Model）
 - 這樣的模型同Keras的Sequential模型一樣，都可以被訓練

```
from keras.layers import Input, Dense
from keras.models import Model

# 這部分返回一個張量
inputs = Input(shape=(784,))

# 層的實例是可調用的，它以張量為參數，並且返回一個張量
x = Dense(64, activation='relu')(inputs)
x = Dense(64, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

# 這部分創建了一個包含輸入層和三個全連接層的模型
model = Model(inputs=inputs, outputs=predictions)
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.fit(data, labels) # 開始訓練
```



解題時間

It's Your Turn

請跳出PDF至官網Sample Code & 作業
開始解題

