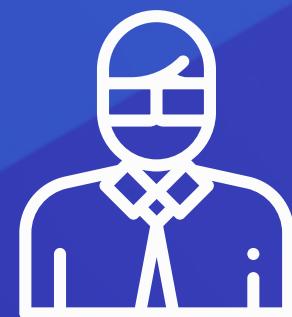


Day 89

初探深度學習使用 Keras

訓練神經網路的細節與技巧 撰寫自己的 Loss function



游為翔

出題教練

知識地圖 深度學習訓練技巧

撰寫自己的 Loss function

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

深度學習訓練技巧

Training Skill of DNN

應注意的關鍵

防止過擬合 (Overfitting)

超參數 (Hyper-parameters)

學習率 (Learning Rate) 調整

相關訓練技巧

正規化
Regularization

批次標準化
Batch Normalization

回呼
Callback

隨機移除
Drop out

客製化損失函數
Customized Loss Function

提前終止
Early Stopping

本日知識點目標

- 學會如何使用自定義的損失函數

Custom loss function in Keras

- 在 Keras 中，除了使用官方提供的 Loss function 外，亦可以自行定義/修改 loss function
- 所定義的函數
 - 最內層函式的參數輸入須根據 output tensor 而定，舉例來說，在分類模型中需要有 y_{true} , y_{pred}
 - 需要使用 tensor operations – 即在 tensor 上運算而非在 numpy array 上進行運算
 - 回傳的結果是一個 tensor

Custom loss function in Keras

```
import keras.backend as K

def dice_coef(y_true, y_pred, smooth):
    y_pred = y_pred >= 0.5
    y_true_f = K.flatten(y_true)
    y_pred_f = K.flatten(y_pred)    皆須使用 tensor operations
    intersection = K.sum(y_true_f * y_pred_f)

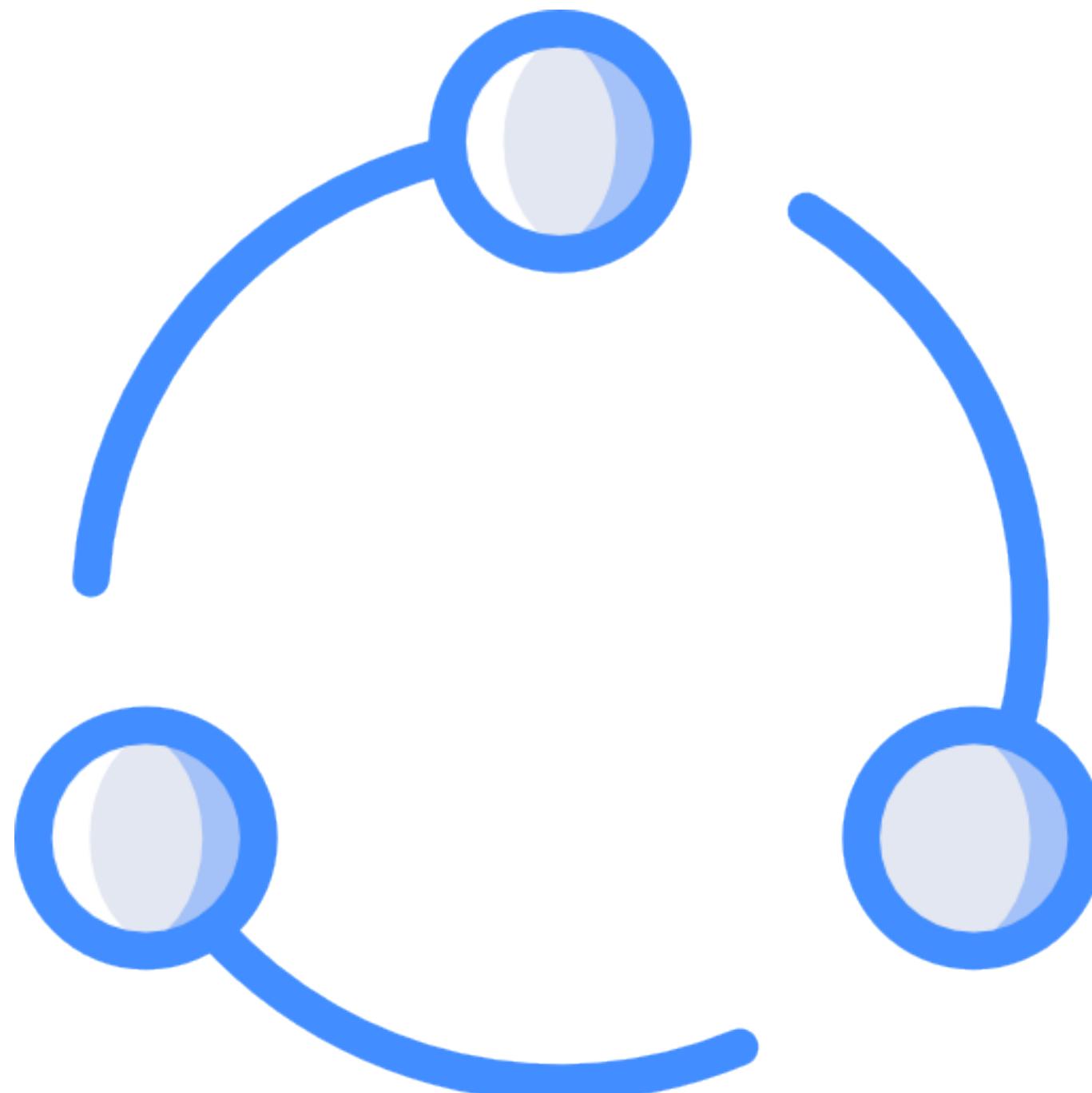
    return (2. * intersection + smooth) /
           (K.sum(y_true_f) + K.sum(y_pred_f) + smooth)

def dice_loss(smooth, thresh):
    def dice(y_true, y_pred):
        return -dice_coef(y_true, y_pred, smooth, thresh)
    return dice
```

輸出為 Tensor

最內層的函式 – 在分類問題中，只能有
y_true 與 y_pred，其他調控參數應至於外層函式

重要知識點複習：



- 在 Keras 中，我們可以自行定義函式來進行損失的運算。一個損失函數必須：
 - 有 `y_true` 與 `y_pred` 兩個輸入
 - 必須可以微分
 - 必須使用 tensor operation，也就是在 tensor 的狀態下，進行運算。如 `K.sum` ...



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

CSDN - Keras 自定義 Loss 函數

有時候我們想根據任務調整損失函數，Keras 可以在 compile model 時使用自定義函數。

最常用的方式

- 自定義函數必須至少要有兩個參數：y_true, y_pred。其他參數則可視狀況自行加入。

較不常用的方式

- 定義一個 loss_layer
- 在 call function 中用 self.add_loss 加其加入

參考來源

```
#custom loss
def mycrossentropy(y_true, y_pred, e=0.1):
    return (1-e)*K.categorical_crossentropy(y_pred,y_true) + \
           e*K.categorical_crossentropy(y_pred, K.ones_like(y_pred)/num_classes)

model.compile(loss=mycrossentropy,
               optimizer=keras.optimizers.Adadelta(),
               metrics=['accuracy'])

class CustomVariationalLayer(Layer):
    def __init__(self, **kwargs):
        self.is_placeholder = True
        super(CustomVariationalLayer, self).__init__(**kwargs)

    def vae_loss(self, x, x_decoded_mean):
        xent_loss = original_dim * metrics.binary_crossentropy(x, x_decoded_mean)#Square Loss
        kl_loss = - 0.5 * K.sum(1 + z_log_var - K.square(z_mean) - K.exp(z_log_var), axis=-1)
        return K.mean(xent_loss + kl_loss)

    def call(self, inputs):
        x = inputs[0]
        x_decoded_mean = inputs[1]
        loss = self.vae_loss(x, x_decoded_mean)
        self.add_loss(loss, inputs=inputs)
# We won't actually use the output.
        return x
```



解題時間

It's Your Turn

請跳出PDF至官網Sample Code & 作業
開始解題

