

人工智慧與深度學習基礎與應用 (TensorFlow 與 Keras) (5)



深度學習的難題



課程範圍

一、提升CNN模型訓練的準確率

- ◆ 訓練資料增強法
- ◆ CNN 網路模型再利用
 - ✓ 特徵擷取法
 - ✓ 基底(CNN Base)再利用法

二、Google CoLab (Colaboratory) 運用

- ◆ 功能介紹
- ◆ Git Hub 結合應用

三、實例練習



一、提升CNN訓練模型的準確率

- ◆ 訓練資料增強法
- ◆ CNN 網路模型再利用
 - ✓ 特徵擷取法
 - ✓ 基底(CNN Base)再利用法



訓練資料增強法(Data Augmentation)

背景說明

深度學習之於影像辨識，一般都採用CNN網路模型作為訓練的方法，CNN的主要特點便是利用卷積演算在物體影像中擷取特徵；而要能夠擷取足夠特徵作為辨識，就需要有大量且有意義的資料來支持。對於物體的辨識，在訓練時的所需收集之圖片，除了數量必須夠多，而並須廣泛的收集此一物體於各種的環境下顯示的形狀（如不同的物體變形），以及光線明暗度所呈現的型態；如此才能夠使得產生的訓練模型於推論有高度的準確率。反之，進行深度學習訓練時，如果資料量不足或資料近似度過多，就會產生過度擬合（over-fitting）的現象，影響推論的準確度。

對於大公司而言，如 Google、Apple、Amazon 等等，收集幾百萬張圖片作為訓練超大規模的深度學習模型，自然相對容易；但是對於個人或者小型公司而言，收集真實的資料，並且需有帶標籤的資料，著實是一件非常艱鉅的工作，在沒有足夠訓練資料的情況之下，資料增強就是在深度學習中不可或缺的運用方法。



Keras 影像圖片轉換函式

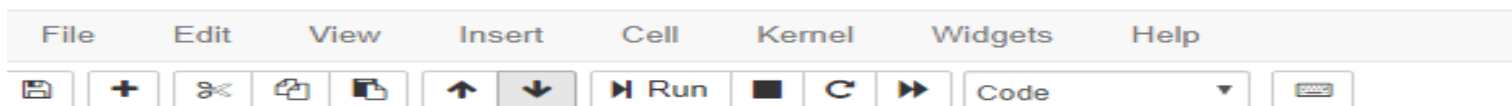
```
import keras
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=60,      # 旋轉角度
    width_shift_range=0.2,  # 寬度平移比例
    height_shift_range=0.2, # 高度平移比例
    shear_range=0.2,        # 變形量
    zoom_range=0.2,         # 縮放比率
    horizontal_flip=True,    # 翻轉
    fill_mode='nearest')    # 圖片空白填補方式
```



Keras 訓練資料增強法範例

jupyter Training_Data_Augmentation_20190725 Last Checkpoint: 5 /



```
In [1]: import keras
keras.__version__
import os, shutil
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
# This is module with image preprocessing utilities
from keras.preprocessing import image

C:\USERS\SHIHCHIN\ANACONDA3\ENV\TENSORFLOW\lib\site-packages\
ment of issubdtype from `float` to `np.floating` is deprecated
t).type`.
    from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

```
In [2]: base_dir = 'D:\cats and dogs small'
# Directory with our training cat pictures
train_dir = os.path.join(base_dir, 'train')
train_cats_dir = os.path.join(train_dir, 'cats')
```

原始訓練資料放置位址，由
Kaggle網址擷取貓狗圖片資料

```
In [3]: datagen = ImageDataGenerator(
    rotation_range=60,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

使用Keras API擴增原始訓練資料 - 宣告資料擴增方式



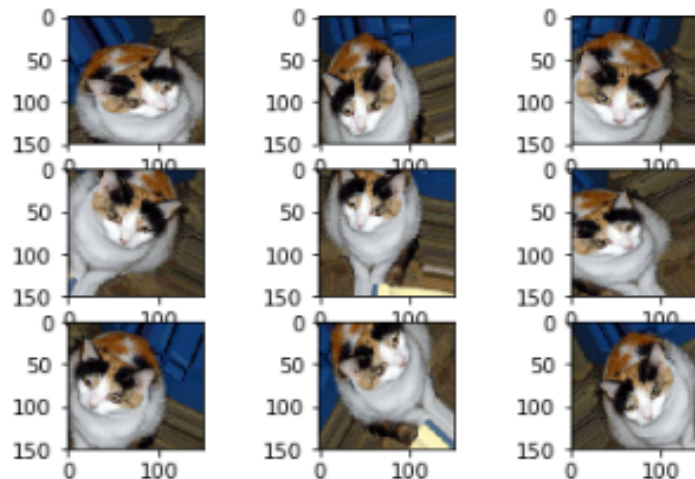
```

In [27]: fnames = [os.path.join(train_cats_dir, fname) for fname in os.listdir(train_cats_dir)]
# We pick one image to "augment"
img_path = fnames[15]
# Read the image and resize it
img = image.load_img(img_path, target_size=(150, 150))
# Convert it to a Numpy array with shape (150, 150, 3)
x = image.img_to_array(img)
# Reshape it to (1, 150, 150, 3)
x = x.reshape((1,) + x.shape)
i = 0
fig = plt.figure()
for batch in datagen.flow(x, batch_size=1):
    sub_img = fig.add_subplot(331 + i)
    sub_img.imshow(image.array_to_img(batch[0]))
    i += 1
    if i % 9 == 0:
        break
plt.show()

```

載入資料

進行資料擴增方式

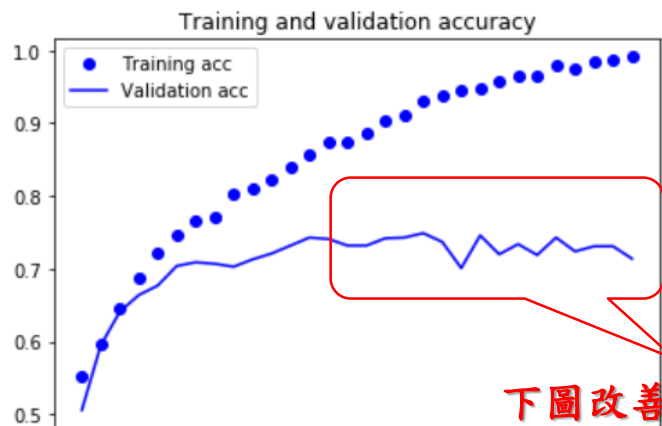


共顯示9張圖



keras 資料增強法效能比較

- ◆ 無資料增強 (tra_acc: 0.9915/loss: 0.0386 , val_acc: 0.7140/val_loss: 1.0472)

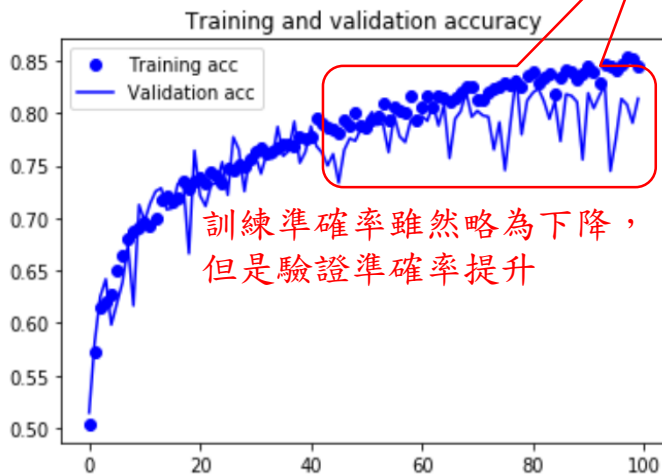


下圖改善
Overfitting現象

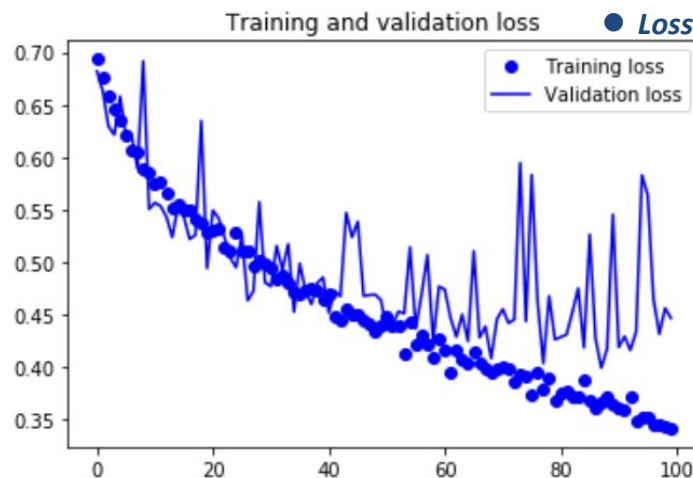
Note:

- Validation Accuracy Increase
- Loss Decrease

- ◆ 資料增強 (tra_acc: 0.8441/loss: 0.3437 , val_acc: 0.8138/val_loss: 0.4466)



訓練準確率雖然略為下降，
但是驗證準確率提升



準確率

損失函數



CNN 網路模型再利用

背景說明(1)

使用CNN為基礎的深度學習模型，在近年來國際競賽中如雨後春筍般的竄出，而準確率也都比往年進步。除此之外，各類型的免費訓練資料也開始出現在網路上，並垂手可得，如 ImageNet 網站(為史丹佛李飛飛所提出的計畫)，就提供數百樣類型的圖片可供下載，作為訓練資料使用。

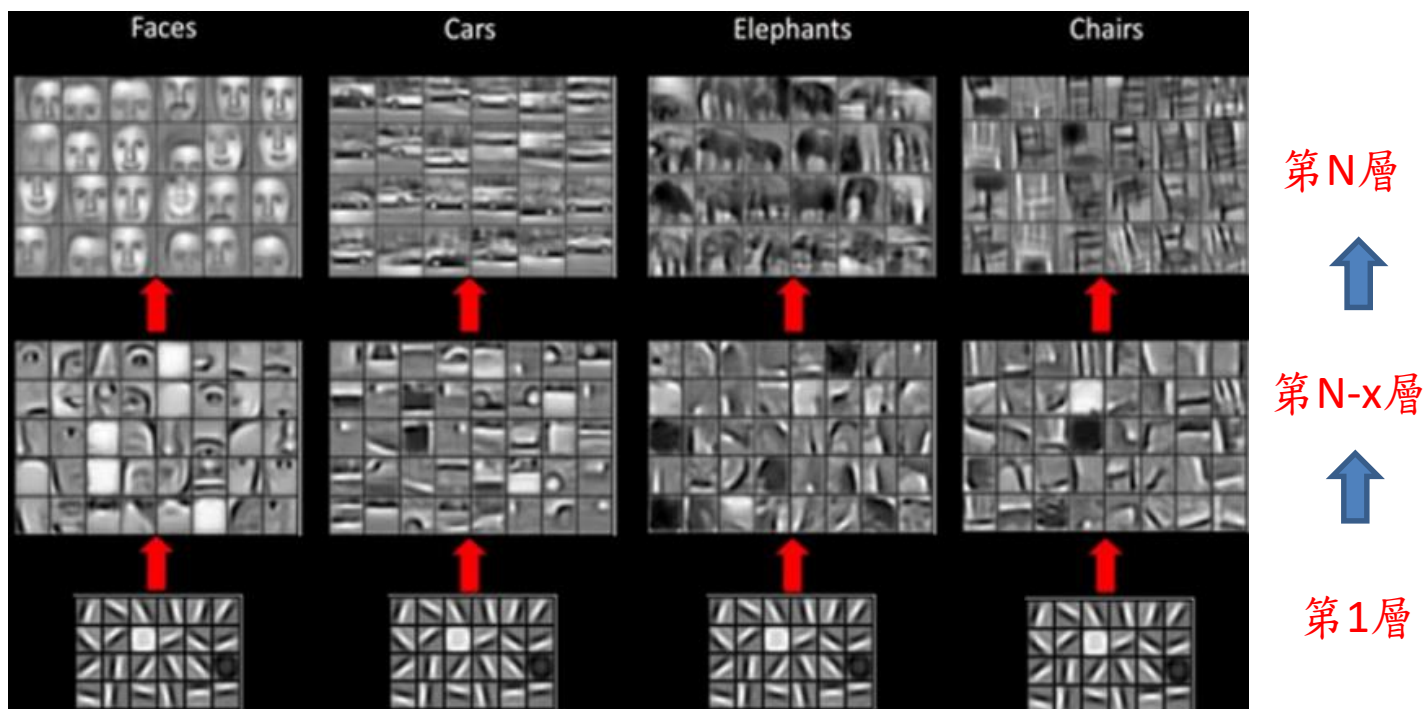
如 2014 年，GoogLeNet 和 VGG 是參與 ImageNet 挑戰賽 (ILSVRC14) 的前兩名：GoogLeNet 第一、VGG 第二；這兩種網路模型結構的共同特點是使用 CNN 架構，且深度學習的隱藏層次很多，GoogLeNet 深度多達 22 層，VGG 繼承了 LeNet 以及 AlexNet 的一些框架結構，VGG16 有 16 層，VGG19 則是 19 層。以模型結果比較來看，GoogLeNet 的效能較好。

以上這些著名且具有公信準確率的網路模型，都具有足夠層次的深度，且經由長時間的訓練，也必須經過大量資料來支持其訓練。這些網路模型是否可以再利用？以 CNN 而言，答案是肯定的。基本上，網路模型再利用的方式可歸納為兩類：特徵擷取法，與基底(CNN Base)再利用(或重建)之方法。



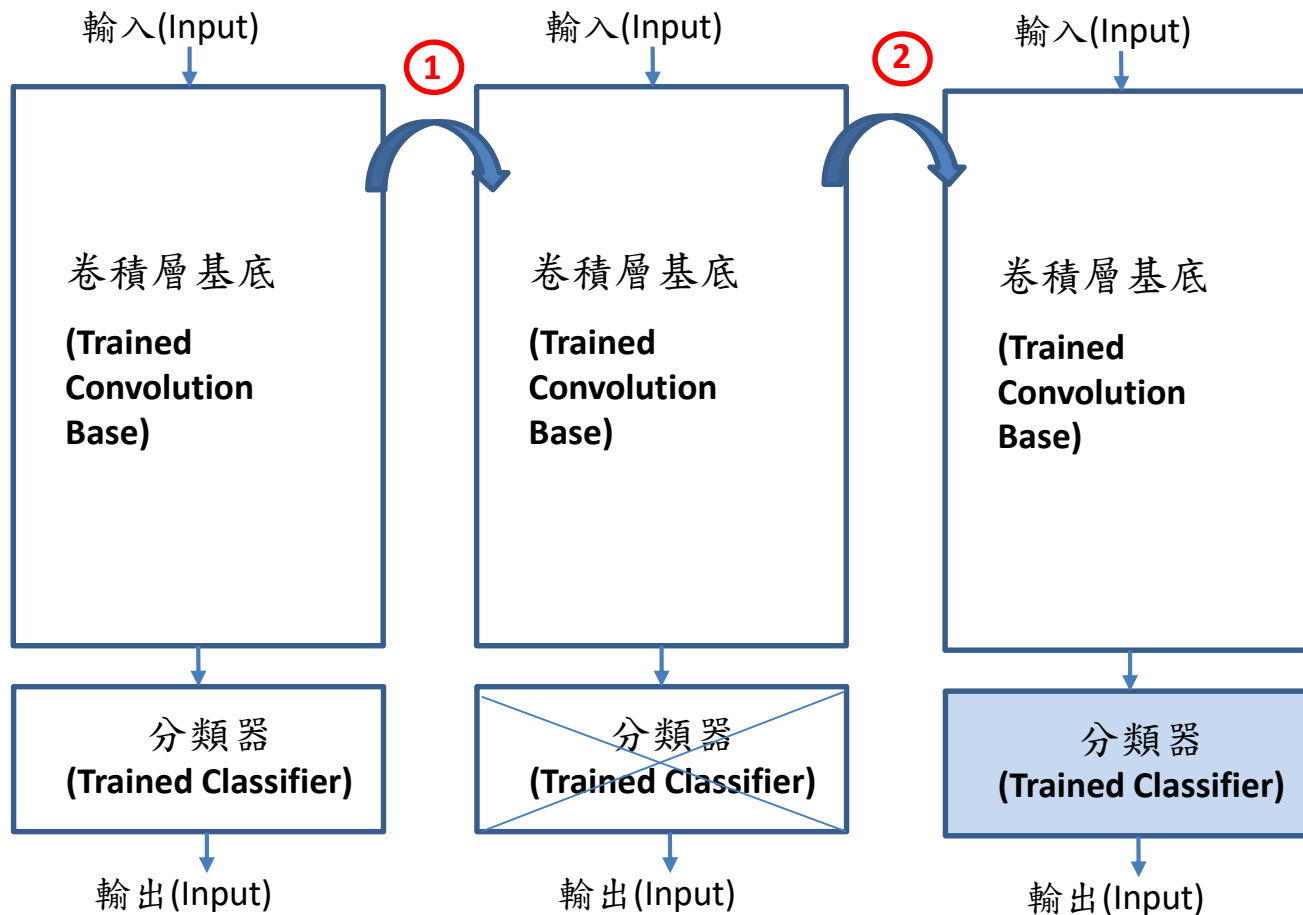
背景說明(2)

以 CNN 各個網路層次而言，基礎層(較接近原始資料輸入的層次)的輸出是具有物體圖片的紋理、線條及顏色等特徵。而較後面的層次偏向萃取抽象的概念，可藉以辨識接近實際物體形狀，如下圖。CNN 模型的再利用，**是以基礎層為主**，因為圖形特徵共用性較明顯，而後面的層次只著重物體分類效果，如密集層，則是完全沒有再利用價值，參考以下圖示。



特徵擷取法(Feature Extraction)

使用訓練成熟的卷基層基底(如VGG16)產生特徵，再輸入至以新的分類器(取代原分類器)重新訓練



Keras 特徵擷取法範例(1)

VGG 深度學習網路簡介

VGG 是以CNN 深度學習為基礎的網路模型。是英國牛津大學 Visual Geometry Group 的縮寫，主要貢獻是使用更多的隱藏層(窄而深)，大量的圖片訓練，**提高準確率至90%**。VGG16/VGG19分別為16層(13個卷積層及3個全連接層)與19層(16個卷積層及3個全連接層)，

VGG16 模型結構說明

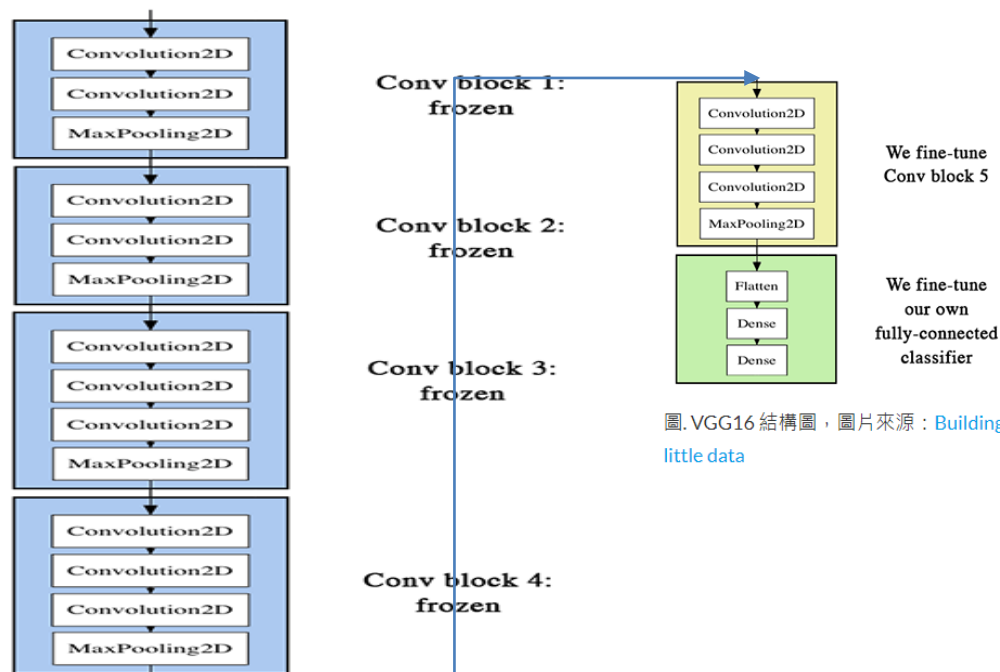


圖. VGG16 結構圖，圖片來源：[Building powerful image classification models using very little data](#)



Keras 特徵擷取法範例(2)

VGG16 特徵擷取之方法步驟：

步驟 1，準備所需要的訓練目標圖形資料 (.jpg/.png 等)，可至 Kaggle 網站下載，如辨識貓、狗圖片；準備各類近似相關圖片。

步驟 2，使用 Keras API 引用 VGG16 CNN 基底訓練以上目標資料。

步驟 3，運用 VGG16 基底，使用 Keras API 抓取原網路模型特徵資料。

步驟 4，建立一個含密集層的網路模型架構作為分類器 (Classifier)，(如辨識貓、狗之分類器)。

步驟 5，將抓取的特徵資料輸入新模型架構中(步驟 4)，以產生新的 CNN 網路模型。



Keras 特徵擷取法範例(3)

程式範例說明

jupyter 5.3_20190731-using-a-pretrained-convnet Last Checkpoint:

File Edit View Insert Cell Kernel Widgets Help

```
In [1]: import keras
keras.__version__
from keras.applications import VGG16
conv_base = VGG16(weights='imagenet',
                    include_top=False,
                    input_shape=(150, 150, 3))
conv_base.summary()
```

使用VGG16模型

```
In [4]: import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'c:/Users/jimli/Desktop/Master-deep-learning-with-python-notebooks/cats_and_dogs_small'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

datagen = ImageDataGenerator(rescale=1./255)
batch_size = 20

def extract_features(directory, sample_count):
    features = np.zeros(shape=(sample_count, 4, 4, 512))
    labels = np.zeros(shape=(sample_count))
    generator = datagen.flow_from_directory(
        directory,
        target_size=(150, 150),
        batch_size=batch_size,
        class_mode='binary')
    i = 0
    for inputs_batch, labels_batch in generator:
        features_batch = conv_base.predict(inputs_batch)
        features[i * batch_size : (i + 1) * batch_size] = features_batch
        labels[i * batch_size : (i + 1) * batch_size] = labels_batch
        i += 1
        if i * batch_size >= sample_count:
            # Note that since generators yield data indefinitely in a loop,
            # we must `break` after every image has been seen once.
            break
    return features, labels

train_features, train_labels = extract_features(train_dir, 2000)
validation_features, validation_labels = extract_features(validation_dir, 1000)
test_features, test_labels = extract_features(test_dir, 1000)
```

目標資料

擷取VGG16模型特徵



```
In [5]: train_features = np.reshape(train_features, (2000, 4 * 4 * 512))
validation_features = np.reshape(validation_features, (1000, 4 * 4 * 512))
test_features = np.reshape(test_features, (1000, 4 * 4 * 512))
```

```
In [6]: from keras import models
from keras import layers
from keras import optimizers
```

```
model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=4 * 4 * 512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
model.compile(optimizer=optimizers.RMSprop(lr=2e-5),
              loss='binary_crossentropy',
              metrics=['acc'])
```

```
history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=20,
                    validation_data=(validation_features, validation_labels))
```

僅加入一密集層，
及分類器，利用原
特徵訓練




```
In [7]: model.save('Using_VGG_Pretrain_20190731.model')
```

```
In [9]: import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

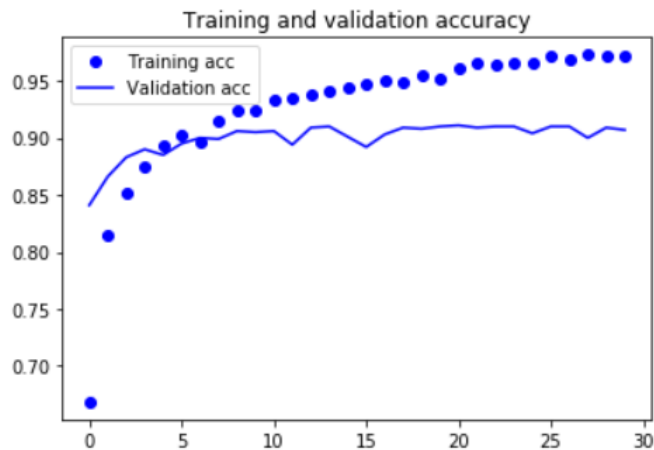
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

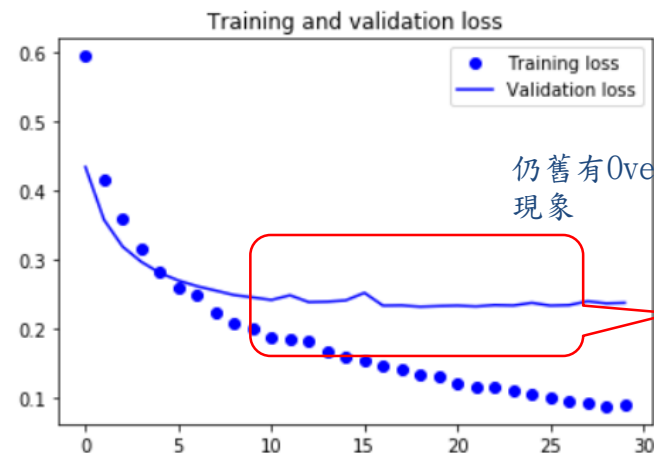
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

◆ 使用特徵擷取 (tra_acc: 0.9725/loss: 0.091 , val_acc: 0.9070/val_loss: 0.2377)



準確率



仍舊有Overfitting的現象

損失函數



Keras 特徵擷取法優缺點(4)

優點：

使用訓練過的VGG16模型，可以獲取並運用VGG16已經訓練的特徵圖，實作在新的目標辨識模型中，讓推論更準確。並且，訓練所需的資料量可以大幅縮減；另外，使用資料少，執行速度非常快。

缺點：

因為使用資料量小，並且不使用資料擴充方法，訓練的成果中可以明顯看出仍有Overfitting的情況，也就是對於同樣或相似於訓練樣本可以有效辨識；但差距較大的測試樣本，無法有效推論。



CNN基底再利用法(Convolution Base Reuse)

方法說明：

以上所提出的特徵擷取法僅須使用少量目標訓練資料，並運用 VGG16 來擷取特徵圖，雖然速度快，但限制是不能合併使用資料擴充法。因為訓練資料有限，訓練成果仍有Overfitting現象。

CNN 基底再利用法則是直接使用 VGG16 基底模型，訓練新資料，同樣的，也替換掉最後一層的分類器。另外，可以搭配資料擴增方法來增加訓練資料，以提高準確率。



Keras 基底再利用之範例(1)

程式範例說明

```
In [ ]: import keras
        keras.__version__
```

```
In [ ]: from keras import models
        from keras import layers
        from keras.applications import VGG16
        conv_base = VGG16(weights='imagenet',
                             include_top=False,
                             input_shape=(150, 150, 3))
        from keras import optimizers

        model = models.Sequential()  # 使用VGG16模型基底
        model.add(conv_base)
        model.add(layers.Flatten())
        model.add(layers.Dense(256, activation='relu'))
        model.add(layers.Dense(1, activation='sigmoid'))
```

```
In [ ]: import os
        import numpy as np
        from keras.preprocessing.image import ImageDataGenerator
        base_dir = '/content/KerasBookApplication/cats_and_dogs_small'
        train_dir = os.path.join(base_dir, 'train')
        validation_dir = os.path.join(base_dir, 'validation')
        test_dir = os.path.join(base_dir, 'test')

        train_datagen = ImageDataGenerator(
            rescale=1./255,
            rotation_range=40,
            width_shift_range=0.2,
            height_shift_range=0.2,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest')

        # Note that the validation data should not be augmented!
        test_datagen = ImageDataGenerator(rescale=1./255)

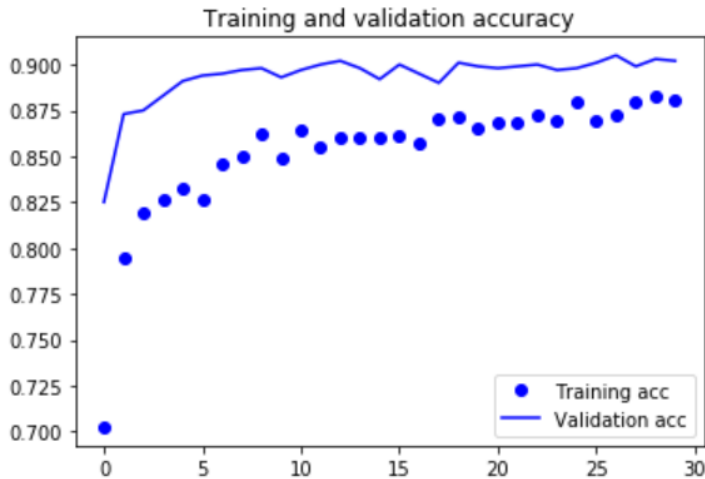
        train_generator = train_datagen.flow_from_directory(
            # This is the target directory
            train_dir,
            # All images will be resized to 150x150
            target_size=(150, 150),
            batch_size=20,
            # Since we use binary_crossentropy loss, we need binary labels
            class_mode='binary')
        validation_generator = test_datagen.flow_from_directory(
            validation_dir,
            target_size=(150, 150),
            batch_size=20,
            class_mode='binary')
        model.compile(loss='binary_crossentropy',
                      optimizer=optimizers.RMSprop(lr=2e-5),
                      metrics=['acc'])
        history = model.fit_generator(
            train_generator,
            steps_per_epoch=100,
            epochs=30,
            validation_data=validation_generator,
            validation_steps=50,
            verbose=2)
```

訓練資料加上資料增強方法，
以VGG16模型為基底重新訓練

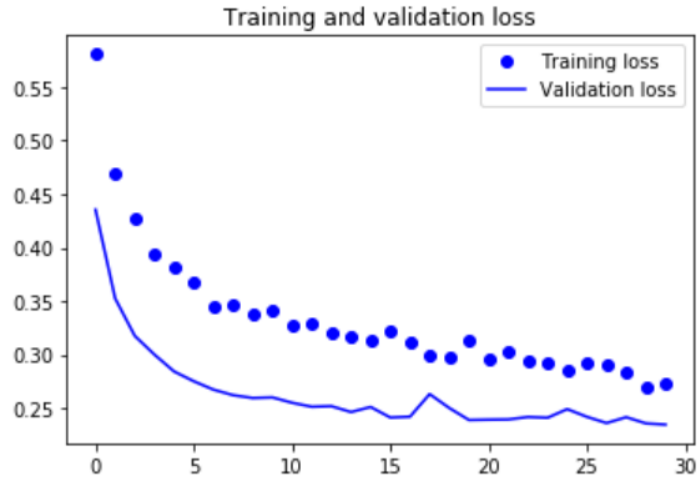


Keras 基底再利用之範例(2)

◆ 使用基底再利用法：(tra_acc: 0.881/loss: 0.2739，val_acc: 0.9020/val_loss: 0.2347)



準確率



損失函數



Keras 基底再利用法優缺點

優點：

驗證準確率 Validation Accuracy 提高 (與特徵擷取比較)，且 Training Accuracy 與 Validation Accuracy 所得到的效果較一致，Validation 甚至比 Training 更好，比較沒有 Overfitting 的問題，實際在推論時效果較佳。

缺點：

因為要訓練大量資料，耗費資源資源，訓練速度慢，實測結果，幾乎無法在僅有 CPU 的筆電電腦上作訓練 (因為每個循環需時數分鐘)。必須使用 GPU/TPU 方能順利訓練，或是利用雲端工具 Cloud Sever 來訓練，如 Google [Colaboratory](#) 的協助 (於主題二介紹)。



二、Google Colab (Colaboratory) 運用

- ◆ Google Colab 功能介紹
- ◆ Git Hub 結合應用



Google Colab 功能介紹

Google Colaboratory（簡稱Google Colab）是 Google 提供的一項雲端服務，供教育或研究領域使用的免費機器學習（Machine Learning）套件；只要有 Google 帳號就可免費使用 Google Colab。Google Colab 類似於網路線上版的 [Jupyter Notebook](#) 使用介面，啟動 Colab 之後就可以開啟直接開始撰寫 Python 3（或Python 2），省下冗長的建置Conda Jupyter Notebook 開發環境時間。

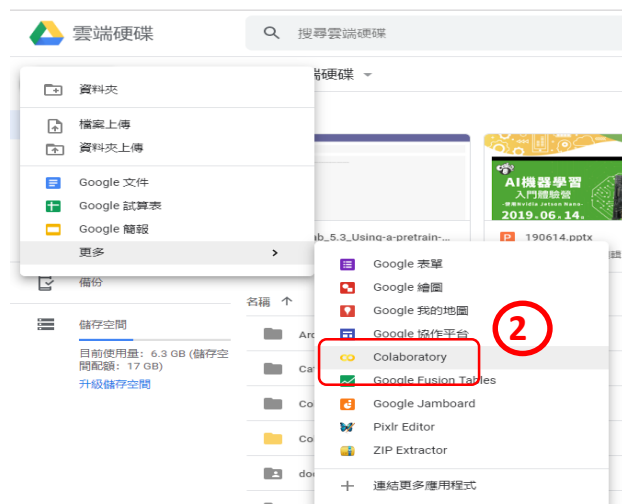
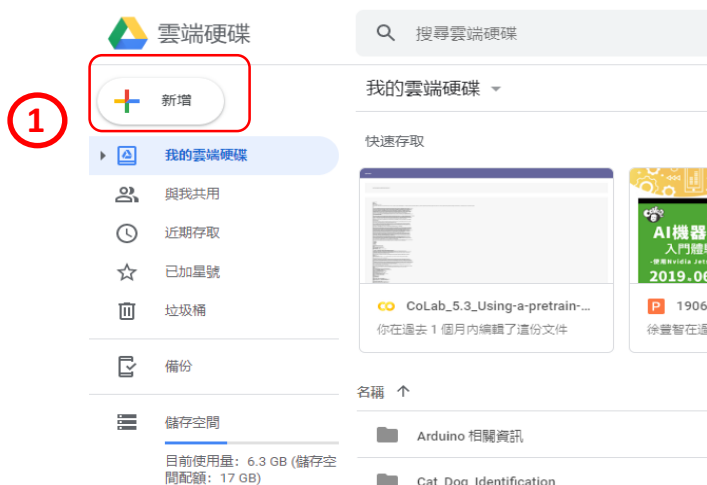
另外，Colab 最重要的強項在於提供的GPU /TPU 運算(GPU 約是 Tesla K80 GPU 等級的效能)，其套件中已經完全安裝 Tensorflow 和 Keras 等常用的深度學習軟件包。因此，如需訓練複雜的網路模型及大量訓練資料，最可行的方法便是上載到 Google Colab 環境上做訓練。

Google CoLab 所提供的環境，又稱為虛擬機，使用者可以同時開啟多的虛擬機。但是，一個虛擬機在使用一段時間後會被自動收回(約12 小時)。虛擬機與Google 雲端硬碟可以結合運用，所以在虛擬機訓練之成果，必須在其時間內保存至雲端硬碟，或可以下載到本機電腦，已備下次訓練之用。



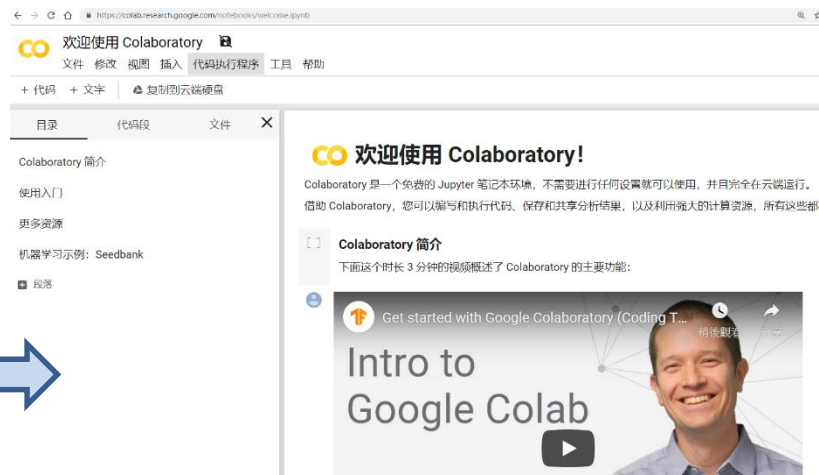
Google Colab 基本操作說明(1)

步驟 1，開啟個人 Google 雲端硬碟 (若無，需申請帳號)。進入雲端硬碟區，按左上角 <新增> 附加功能 <Colaboratory>。



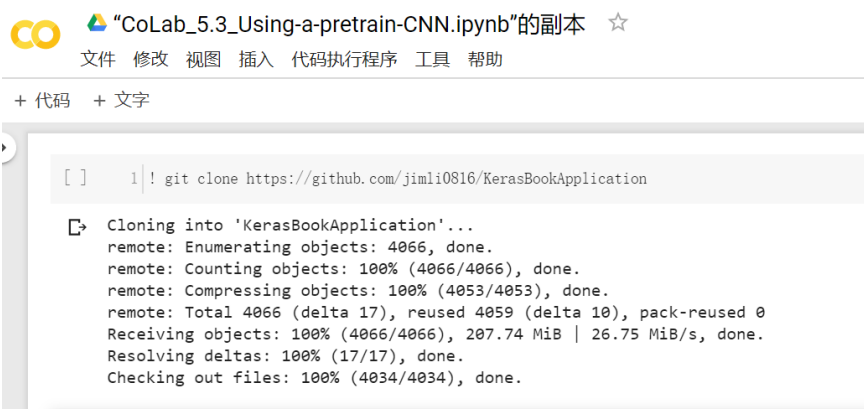
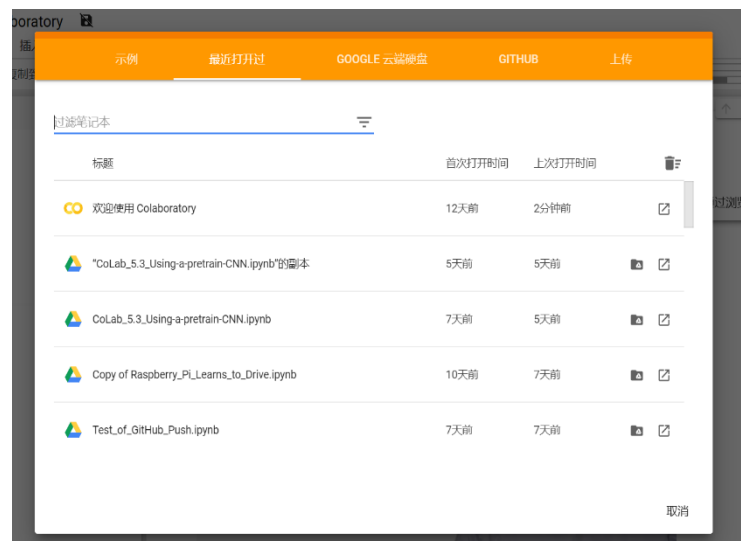
步驟 2，在開啟 Google 瀏覽器，
輸入
<https://colab.research.google.com>
便可使用 Google Colab 虛擬環境

○



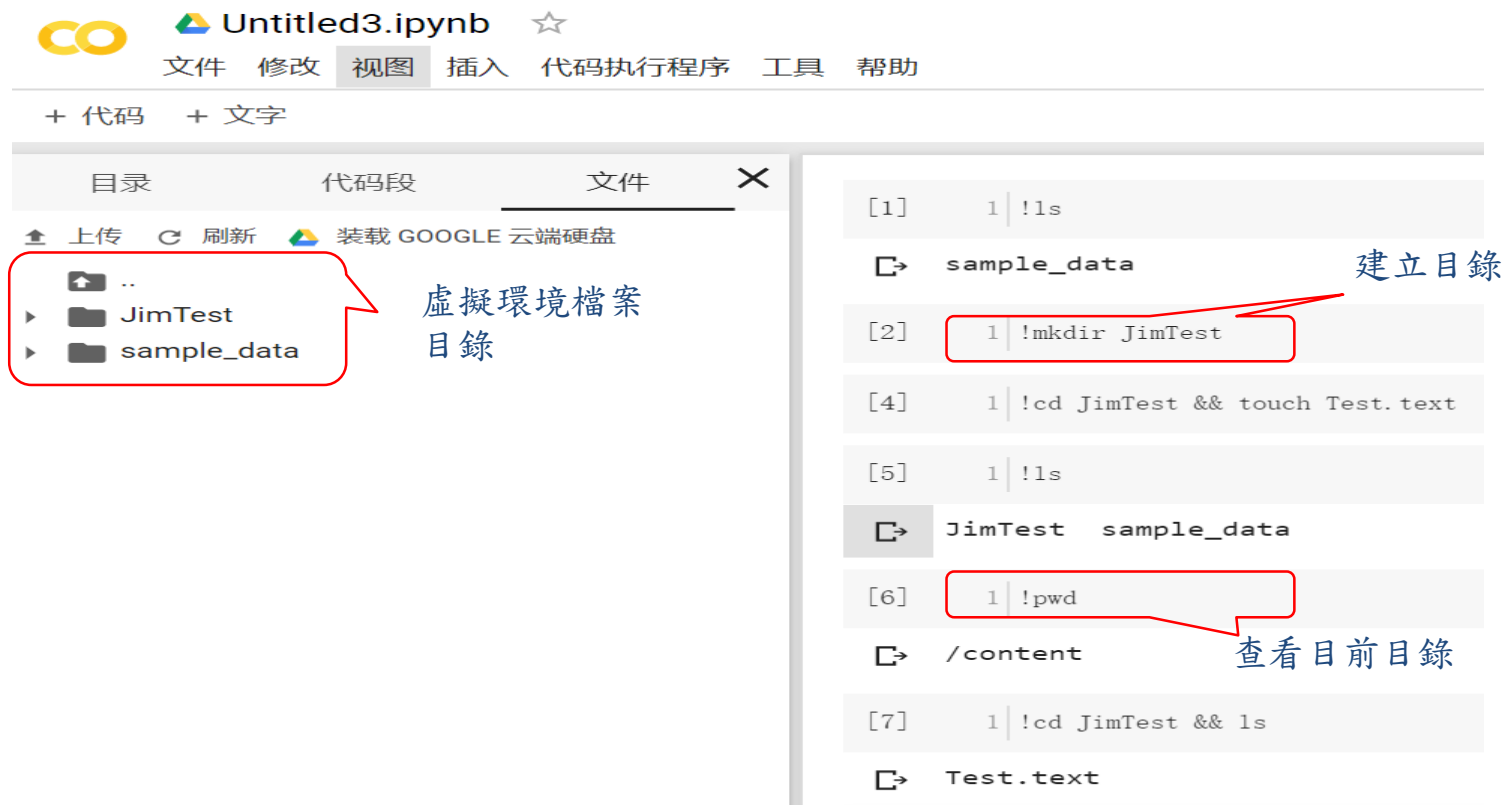
Google Colab 基本操作說明(2)

步驟 3，於文件Menu下，可以創建新的Python程式，或選取原文件所在目錄(雲端硬碟，或本機GitHub檔案，等等)。



Google Colab 基本操作說明(3)

當開啟新的 python3 檔案後，操作環境如同jupyter notebook。而在每一個代碼單元格內除了可寫python 程式外，另外可以以Linux 指令操作檔案及系統功能 (指令字串前加 !)



The screenshot displays the Google Colab interface for a file named 'Untitled3.ipynb'. The top menu bar includes '文件' (File), '修改' (Edit), '视图' (View), '插入' (Insert), '代码执行程序' (Code execution), '工具' (Tools), and '帮助' (Help). Below the menu, there are buttons for '+ 代码' (Code) and '+ 文字' (Text).

On the left, the '文件' (Files) tab is active, showing a file explorer. It includes buttons for '上传' (Upload), '刷新' (Refresh), and '装载 GOOGLE 云端硬盘' (Load Google Drive). The file list shows '..' (parent directory), 'JimTest', and 'sample_data'. A red box highlights the 'JimTest' and 'sample_data' folders, with a callout pointing to them that says '虛擬環境檔案目錄' (Virtual environment file directory).

On the right, the code execution area shows a series of commands in numbered cells:

- [1] `!ls`
- [2] `!mkdir JimTest` (A red box highlights this command, with a callout pointing to it that says '建立目錄' (Create directory).)
- [4] `!cd JimTest && touch Test.text`
- [5] `!ls`
- [6] `!pwd` (A red box highlights this command, with a callout pointing to it that says '查看目前目錄' (View current directory).)
- [7] `!cd JimTest && ls`

Below the code cells, the file explorer shows the current directory structure: `sample_data`, `JimTest`, and `Test.text`.



Google Colab 基本操作說明(4)

在虛擬環境下執行深度學習訓練，可以適時選擇所要使用的硬體加速器類型：GPU或 Google專屬的TPU，以加速訓練。



笔记本设置

运行时类型
Python 3

硬件加速器
GPU

☐ 保存此笔记本时忽略输出项

None

GPU

TPU

取消

保存

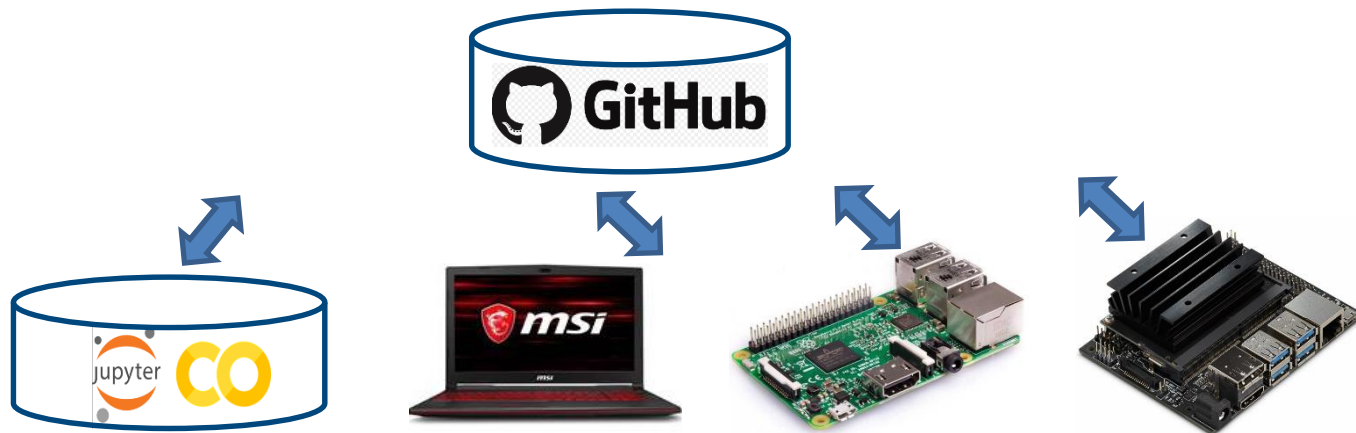
註：實測在深度學習CNN訓練，對於100 Samples/epoch 訓練，筆電 Intel i5 約需 700s，而Google Colab GPU 虛擬環境僅需 30s。



Google Colab 與 GitHub 結合應用

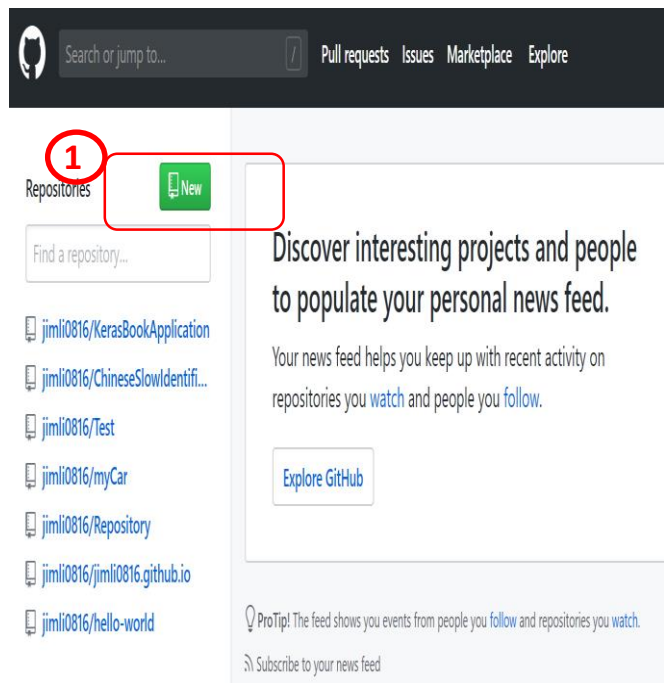
Google Colab 可以滿足一般小企業或Maker 在執行深度學習計畫的需求。如果沒有Google Colab，使用者在執行專案，僅能考慮自行建置昂貴的 Server，或者申請付費的雲端服務。

Google Colab 最大的缺點是無法提供永久的儲存空間。雖然可以搭配使用Google Drive 雲端硬碟，但必須先將本機檔案傳送至Google Drive，再上載至Google Colab，在操作程序上十分繁瑣；如有大量的訓練圖片，也需與程式一起，分別由本機載入Google Colab上的虛擬環境。此外，Google Colab 訓練完成的模型，如提供RaspberryPi 或JetsonNano等嵌入式裝置做推論使用，也需將模型傳至Google Drive 或本機，再轉傳至嵌入式裝置，十分不便。所幸，Google Colab 能與GitHub結合，以提供一種解決檔案傳遞過程中的問題，最有效的方法。方法架構示意圖如下。



GigHub基本操作說明(1)

1. 建立一新目錄，例如： “Pretain_CNN_Model”



Owner: jimli0816

Repository name *: **2** Pretrain_CNN_Model ✓

Great repository names are short and memorable. Need inspiration? How about [fluffy-octo-telegram](#)?

Description (optional): Pretrain CNN model

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None ⓘ

3 Create repository

jimli0816 / Pretrain_CNN_Model

Unwatch 1

<> Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Security | Insights | Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/jimli0816/Pretrain_CNN_Model.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#)

...or create a new repository on the command line

```
echo "# Pretrain_CNN_Model" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/jimli0816/Pretrain_CNN_Model.git
git push -u origin master
```



GigHub基本操作說明(2)

2. Window CMD 將檔案目錄上傳GitHub

```
>git init  
>git add .  
>git commit -m "first commit"  
>git remote add origin https://github.com/jimli0816/Pretrain_CNN_Model.git  
>git push -u origin master
```

3. Colab 下載 GitHub 檔案

CO Untitled4.ipynb ☆

文件 修改 视图 插入 代码执行程序 工具 帮助

+ 代码 + 文字



執行下載 GitHub
之檔案

```
[1] 1 !git clone https://github.com/jimli0816/Pretrain_CNN_Model
```

```
Cloning into 'Pretrain_CNN_Model'...  
remote: Enumerating objects: 4013, done.  
remote: Counting objects: 100% (4013/4013), done.  
remote: Compressing objects: 100% (4013/4013), done.  
remote: Total 4013 (delta 0), reused 4013 (delta 0), pack-reused 0  
Receiving objects: 100% (4013/4013), 86.09 MiB | 44.57 MiB/s, done.
```

```
1 !git config --global user.email "jimli081601@gmail.com"  
2 !git config --global user.name "jimli0816"
```

上傳檔案至
GitHub

```
[ ] 1 !cd Pretrain_CNN_Model && git add .
```



GigHub基本操作說明(3)

在 Google Colab 中修改 GitHub .ipynb 檔案

The image shows a Google Colab notebook interface with several annotations:

- Annotation 1:** A red circle with the number '1' points to the 'GITHUB' tab in the top navigation bar.
- Annotation 2:** A red circle with the number '2' points to the search input field where 'jimli0816' is entered.
- Annotation 3:** A red box highlights the 'Pretrain_CNN_Model' repository in the search results.
- Annotation 4:** A red box highlights the 'Open notebook...' button next to the selected repository.
- Annotation 5:** A blue arrow points from the 'Pretrain_CNN_Model' repository to the '5.3-Colab_using-a-pretrained-convnet.ipynb' notebook.
- Annotation 6:** A blue arrow points from the '5.3-Colab_using-a-pretrained-convnet.ipynb' notebook to the code editor.

The code editor displays the following Python code:

```
[ ] 1 import keras
    2 keras.__version__

'2.2.4'

1 from keras.applications import VGG16
2 conv_base = VGG16(weights='imagenet',
3                       include_top=False,
4                       input_shape=(150, 150, 3))
5

Using TensorFlow backend.
WARNING: Logging before flag parsing goes to stderr
W0731 10:13:01.736836 6396 deprecation_wrapper
W0731 10:13:01.747775 6396 deprecation_wrapper
W0731 10:13:01.749803 6396 deprecation_wrapper
W0731 10:13:01.768752 6396 deprecation_wrapper
```



三、實例練習

- ◆ 收集訓練資料並實作各類深度學習方法
- ◆ Colab 及 Hub 結合運用實例練習
- ◆ 各模型導入嵌入式系統(RaspberryPi 或 Jetson Nano) 推論實作



Enjoy your journey to Deep Learning

You may download programs at : RealPlus's Dropbox

官方網站



LINE



FB

