資料預處理與資料增強

吳庭育

tyw@mail.npust.edu.tw

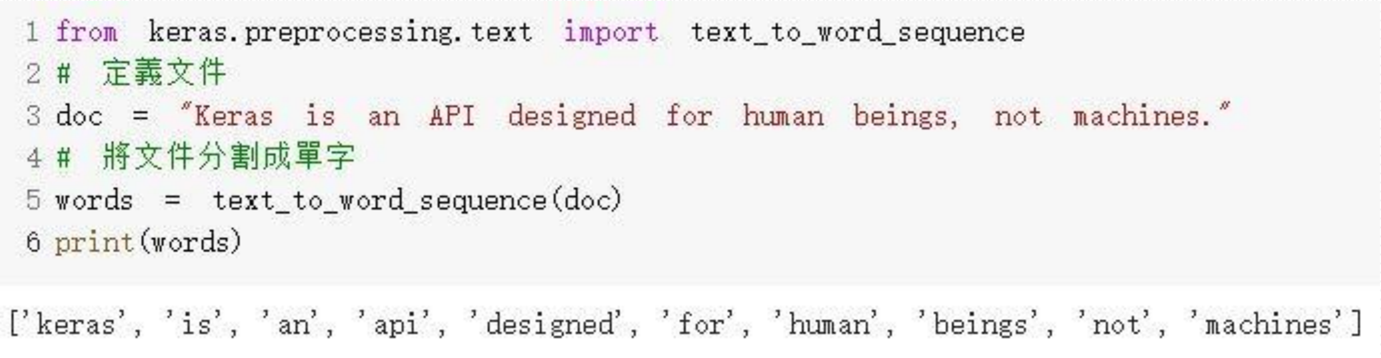
文字資料預處理

分割文字資料 – 斷詞

- Keras的keras.preprocessing.text模組提供相關函式，可以幫助我們進行深度學習模型所需的文字資料預處理。
- 文字資料預處理的第一步是將文字內容分割成一序列的單字(Words)，或稱為Tokens，這種操作稱為「斷詞」(Tokenization)。

分割文字資料

- Keras可以使用text_to_word_sequence()函式將文字內容分割成一序列的單字，將文件的字串分割成英文單字：

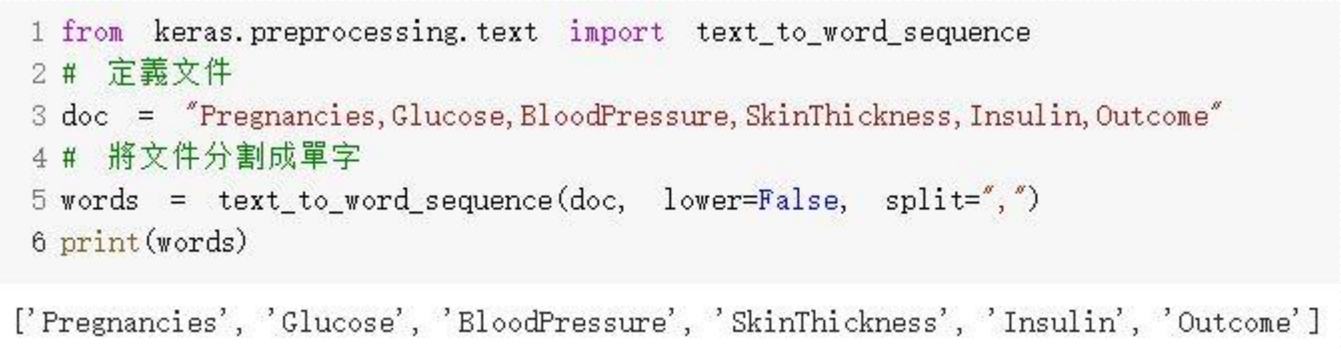


```
from keras.preprocessing.text import text_to_word_sequence
# 定義文件
doc = "Keras is an API designed for human beings, not machines."
# 將文件分割成單字
words = text_to_word_sequence(doc)
print(words)
```

```
['keras', 'is', 'an', 'api', 'designed', 'for', 'human', 'beings', 'not', 'machines']
```

分割文字資料

- 如果不是使用空白字元來分割成單字，我們可以使用split參數定義分割字串的符號，例如：分割「,」符號分隔的字串。



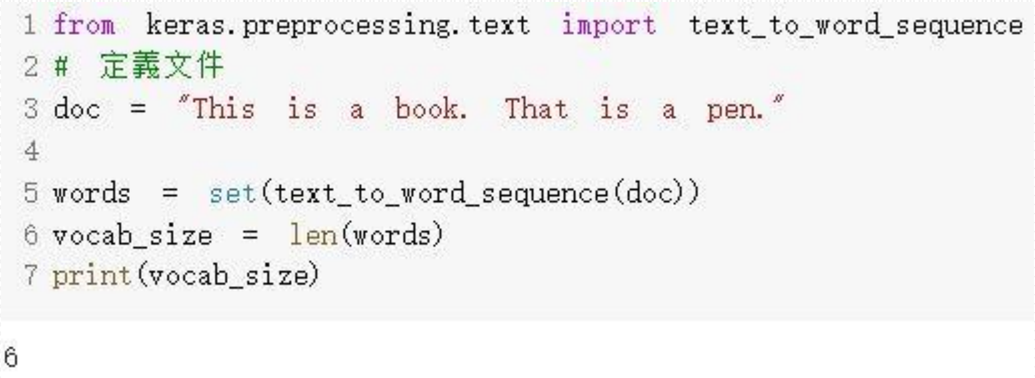
```
from keras.preprocessing.text import text_to_word_sequence
# 定義文件
doc = "Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,Outcome"
# 將文件分割成單字
words = text_to_word_sequence(doc, lower=False, split=",")
print(words)
```

```
['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'Outcome']
```

- text_to_word_sequence()
  - text參數：指定欲分割成單字的字串(第一個參數)。
  - filters參數：指定需要過濾掉那些字元。
  - lower參數：布林值，是否自動轉換成小寫英文字母，預設值為True
  - Split參數：指定分割的字串，預設值是空白字元。

計算文字資料的單字數

- 當成功將文字資料分割成英文單字後，我們就可以使用Python的集合(Set)型別來計算出文字內容的單字數：



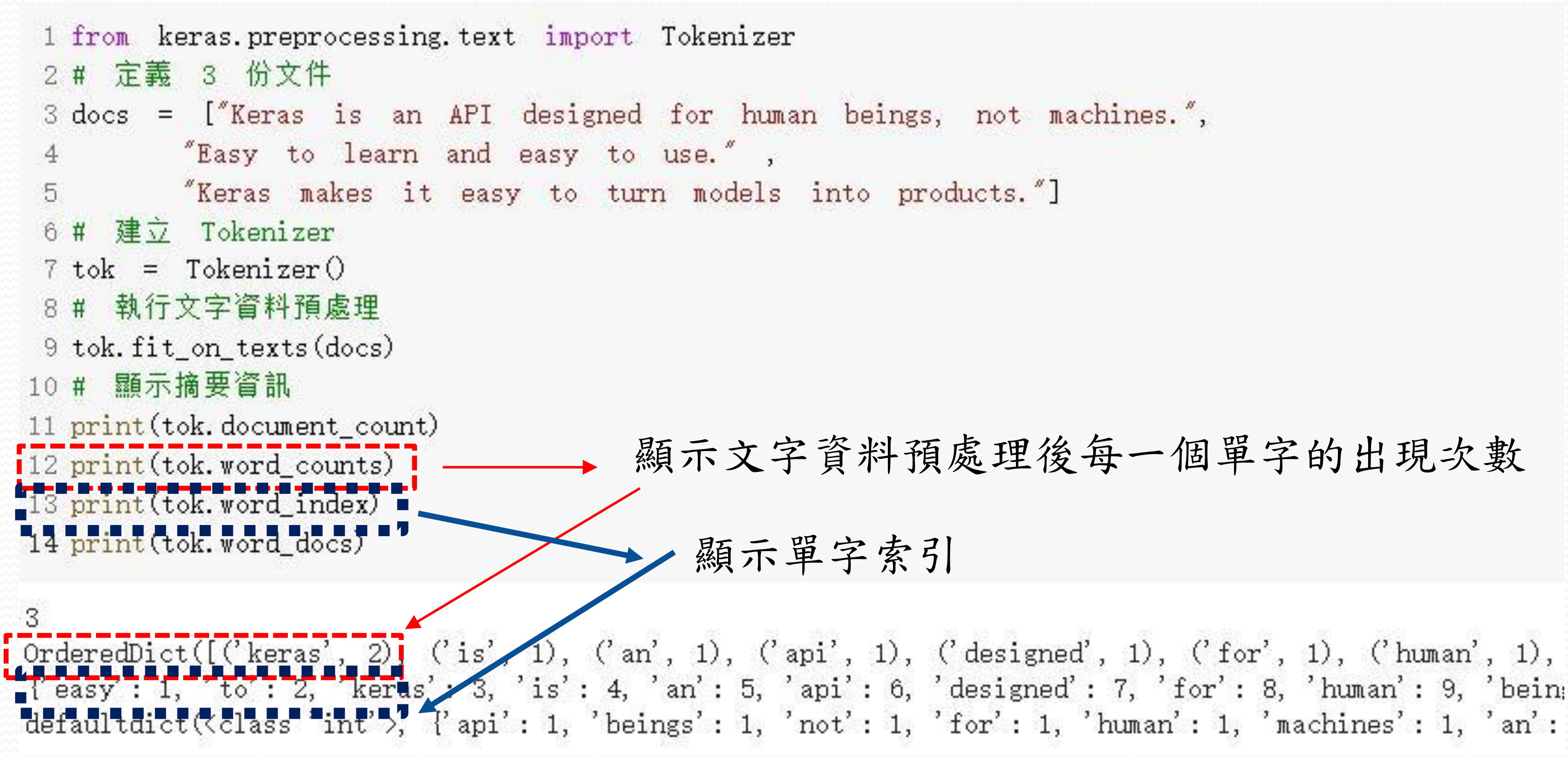
```python
from keras.preprocessing.text import text_to_word_sequence
# 定義文件
doc = "This is a book. That is a pen."

words = set(text_to_word_sequence(doc))
vocab_size = len(words)
print(vocab_size)
```

```
6
```

- 集合中的元素是唯一且不可重複

Tokenizer API -顯示文字資料的摘要資訊

- 匯入Tokenizer物件和定義3份文件的清單



```
from keras.preprocessing.text import Tokenizer
# 定義 3 份文件
docs = ["Keras is an API designed for human beings, not machines.",
       "Easy to learn and easy to use." ,
       "Keras makes it easy to turn models into products."]
# 建立 Tokenizer
tok = Tokenizer()
# 執行文字資料預處理
tok.fit_on_texts(docs)
# 顯示摘要資訊
print(tok.document_count)
print(tok.word_counts)
print(tok.word_index)
print(tok.word_docs)
```

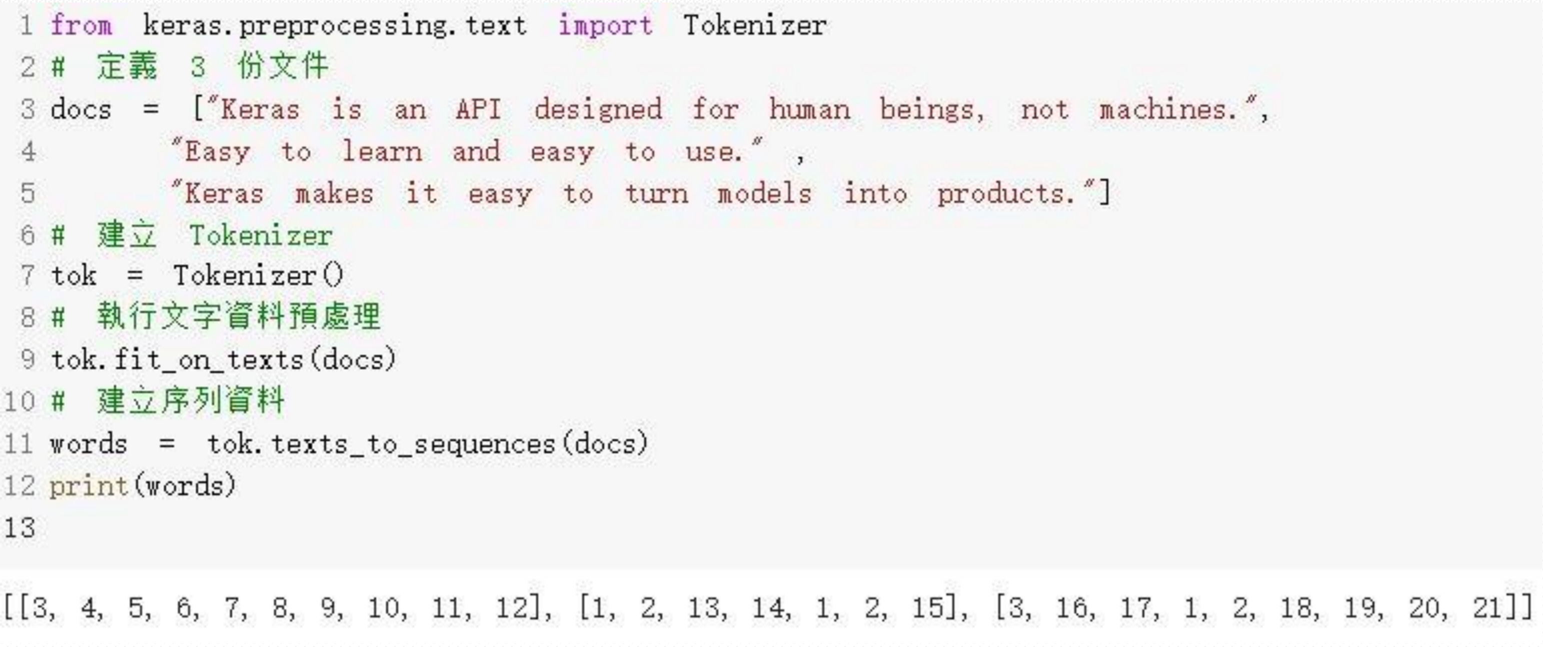
顯示文字資料預處理後每一個單字的出現次數

顯示單字索引

```
3
OrderedDict([('keras', 2), ('is', 1), ('an', 1), ('api', 1), ('designed', 1), ('for', 1), ('human', 1),
{'easy': 1, 'to': 2, 'keras': 3, 'is': 4, 'an': 5, 'api': 6, 'designed': 7, 'for': 8, 'human': 9, 'bein
defaultdict(<class 'int'>, {'api': 1, 'beings': 1, 'not': 1, 'for': 1, 'human': 1, 'machines': 1, 'an':
```

文字資料索引化

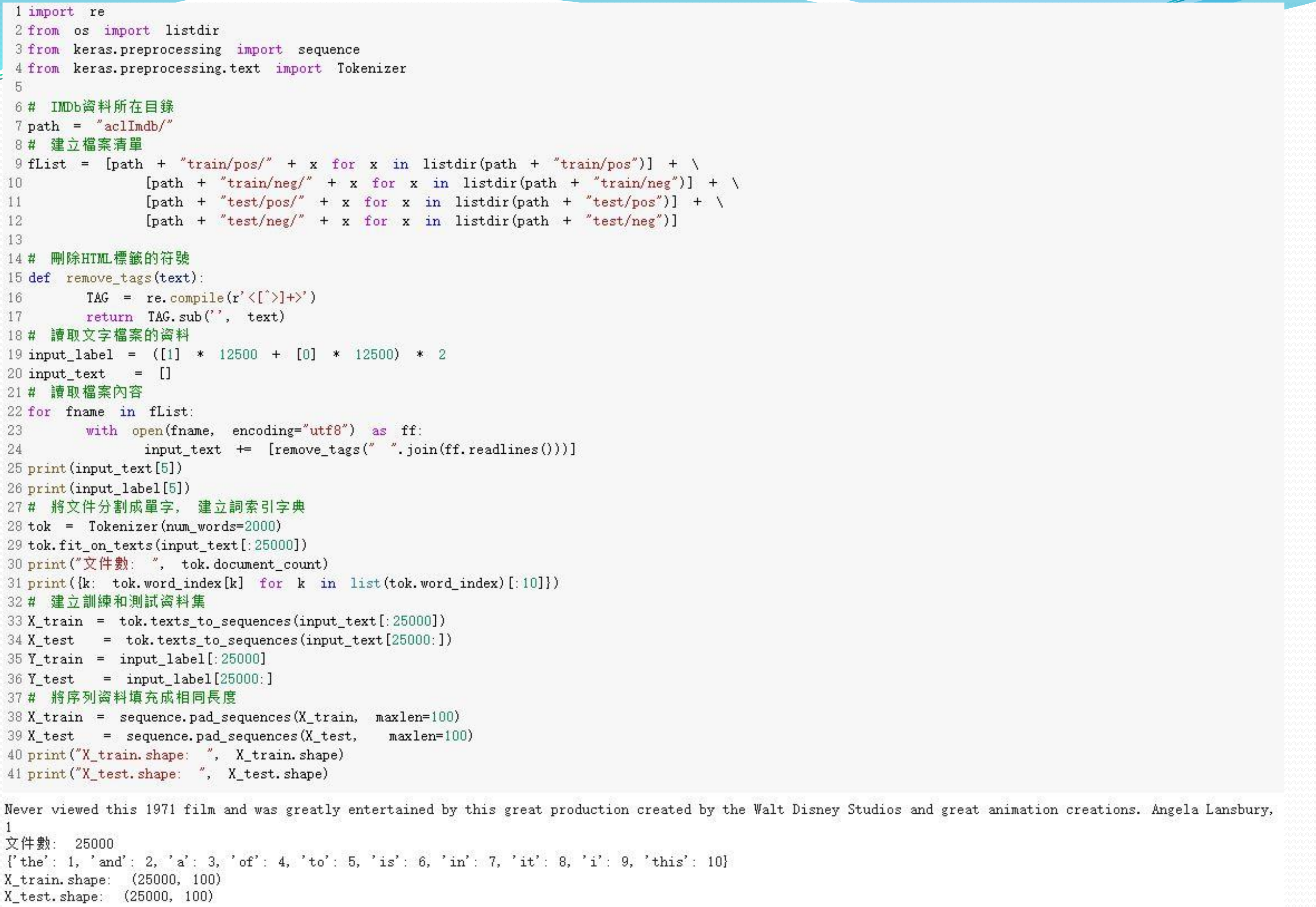
- Tokenizer物件可以將文字資料使用各單字的索引值來執行文字資料索引化：



```
from keras.preprocessing.text import Tokenizer
# 定義 3 份文件
docs = ["Keras is an API designed for human beings, not machines.",
       "Easy to learn and easy to use." ,
       "Keras makes it easy to turn models into products."]
# 建立 Tokenizer
tok = Tokenizer()
# 執行文字資料預處理
tok.fit_on_texts(docs)
# 建立序列資料
words = tok.texts_to_sequences(docs)
print(words)

```

```
[[3, 4, 5, 6, 7, 8, 9, 10, 11, 12], [1, 2, 13, 14, 1, 2, 15], [3, 16, 17, 1, 2, 18, 19, 20, 21]]
```

IMDb網路電影資料預處理



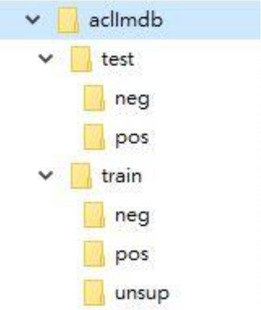
```python
import re
from os import listdir
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer

# IMDb資料所在目錄
path = "aclImdb/"
# 建立檔案清單
fList = [path + "train/pos/" + x for x in listdir(path + "train/pos")] + \
        [path + "train/neg/" + x for x in listdir(path + "train/neg")] + \
        [path + "test/pos/" + x for x in listdir(path + "test/pos")] + \
        [path + "test/neg/" + x for x in listdir(path + "test/neg")]

# 刪除HTML標籤的符號
def remove_tags(text):
    TAG = re.compile(r'<[^>]+>')
    return TAG.sub('', text)
# 讀取文字檔案的資料
input_label = ([1] * 12500 + [0] * 12500) * 2
input_text  = []
# 讀取檔案內容
for fname in fList:
    with open(fname, encoding="utf8") as ff:
        input_text += [remove_tags(" ".join(ff.readlines()))]
print(input_text[5])
print(input_label[5])
# 將文件分割成單字, 建立詞索引字典
tok = Tokenizer(num_words=2000)
tok.fit_on_texts(input_text[:25000])
print("文件數: ", tok.document_count)
print({k: tok.word_index[k] for k in list(tok.word_index)[:10]})
# 建立訓練和測試資料集
X_train = tok.texts_to_sequences(input_text[:25000])
X_test  = tok.texts_to_sequences(input_text[25000:])
Y_train = input_label[:25000]
Y_test  = input_label[25000:]
# 將序列資料填充成相同長度
X_train = sequence.pad_sequences(X_train, maxlen=100)
X_test  = sequence.pad_sequences(X_test,  maxlen=100)
print("X_train.shape: ", X_train.shape)
print("X_test.shape: ", X_test.shape)
```

```
Never viewed this 1971 film and was greatly entertained by this great production created by the Walt Disney Studios and great animation creations. Angela Lansbury,
1
文件數: 25000
{'the': 1, 'and': 2, 'a': 3, 'of': 4, 'to': 5, 'is': 6, 'in': 7, 'it': 8, 'i': 9, 'this': 10}
X_train.shape: (25000, 100)
X_test.shape: (25000, 100)
```

IMDb網路電影資料預處理

- 之前在IMDb情緒分析是使用Keras內建資料集，資料已經索引化，事實上，我們可以自行從原始資料使用Python程式碼處理成情緒分析所需的訓練資料。
  - 下載IMDb網路電影資料
  - IMDb網路電影資料預處理

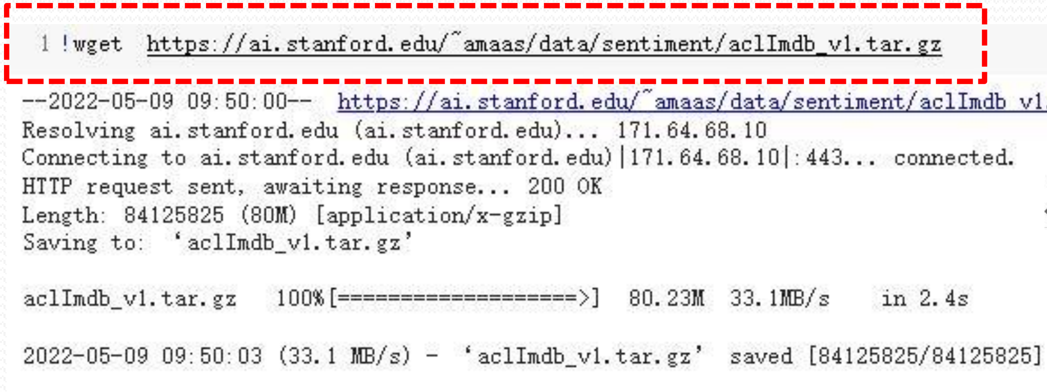
下載IMDb網路電影資料



- aclImdb
  - test
    - neg
    - pos
  - train
    - neg
    - pos
    - unsup

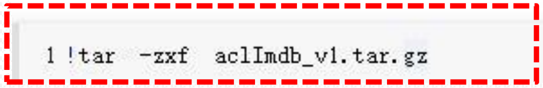
- IMDb網路電影資料的免費下網址：

  https://ai.stanford.edu/~amaas/data/sentiment/
- 請從上述網址下載名為aclImdb_v1.tar.gz的壓縮檔案

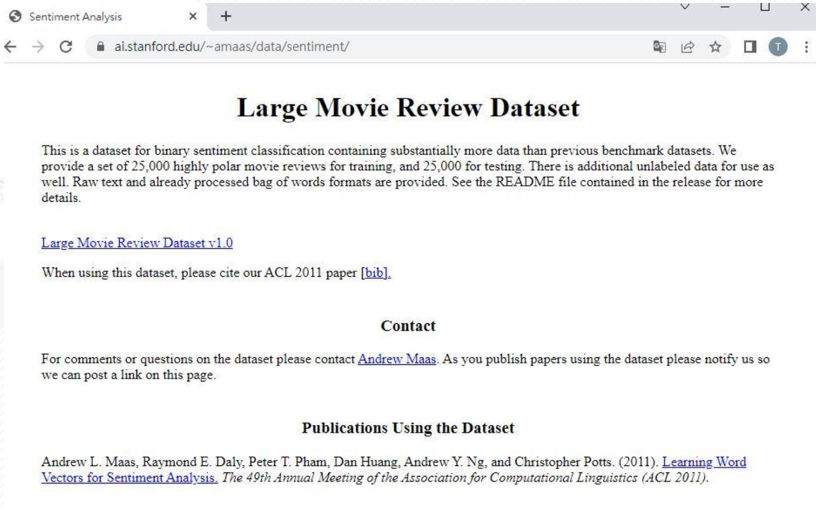


```
!wget  https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
```

```
--2022-05-09 09:50:00--  https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
Resolving ai.stanford.edu (ai.stanford.edu)... 171.64.68.10
Connecting to ai.stanford.edu (ai.stanford.edu)|171.64.68.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 84125825 (80M) [application/x-gzip]
Saving to: ‘aclImdb_v1.tar.gz’

aclImdb_v1.tar.gz   100%[===================>]  80.23M  33.1MB/s    in 2.4s

2022-05-09 09:50:03 (33.1 MB/s) - ‘aclImdb_v1.tar.gz’ saved [84125825/84125825]
```



```
!tar  -zxf  aclImdb_v1.tar.gz
```



Large Movie Review Dataset

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided. See the README file contained in the release for more details.

Large Movie Review Dataset v1.0

When using this dataset, please cite our ACL 2011 paper [bib].

Contact

For comments or questions on the dataset please contact Andrew Maas. As you publish papers using the dataset please notify us so we can post a link on this page.

Publications Using the Dataset

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

IMDb網路電影資料預處理

- 執行IMDb網路電影資料預處理:
  - 首先匯入所需的模組與套件
  - 掃瞄目錄找出所有文字檔案的檔案清單
  - 建立remove_tags()函式刪除HTML標籤的符號
  - 接著建立標籤和評論文字清單
  - 然後開始讀取文字檔案的內容
  - 將文字內容分割成單字來建立詞索引
  - 使用詞索引字典將文字內容轉換成整數清單，即可建立訓練和測試資料集
  - 將序列資料填充成相同長度

圖片載入與預處理

載入圖片檔案

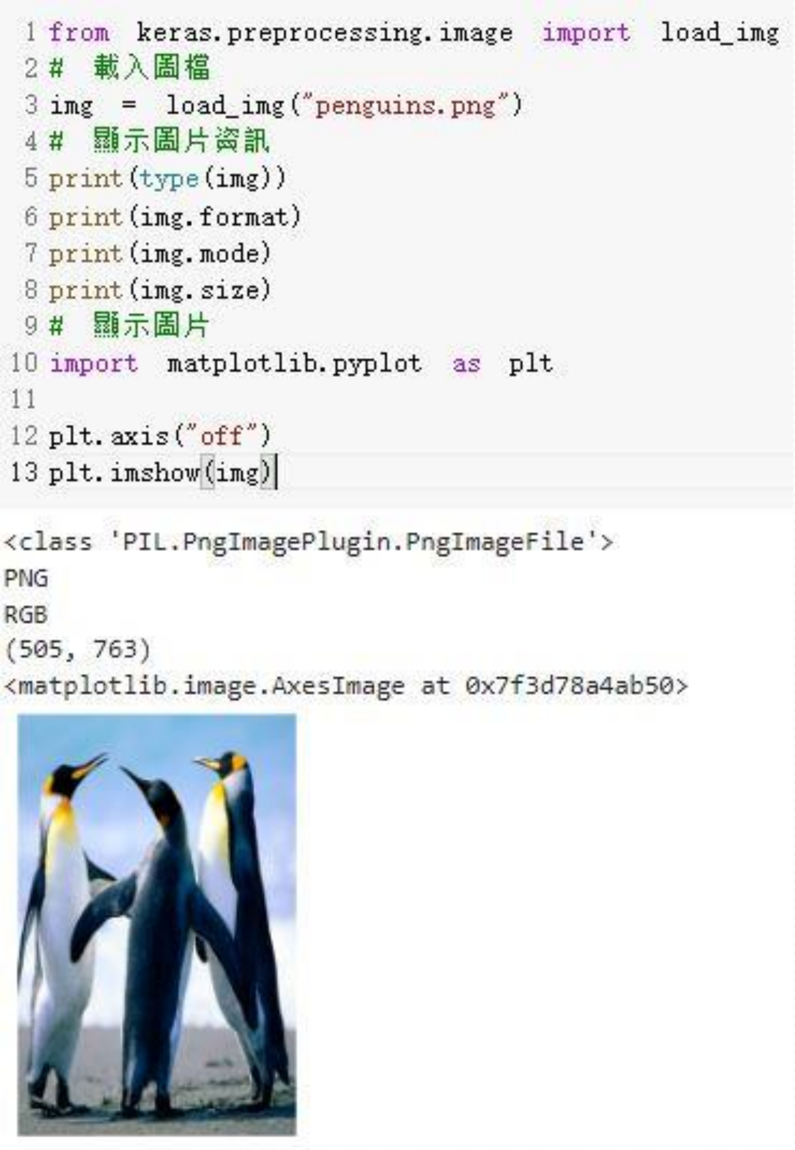
- Keras是使用load_img()函式載入圖檔，首先匯入此函式：

  from keras.preprocessing.image import load_img

- 載入penguins.png的圖檔，建立的是PIL (Python Imaging Library)的PngImageFile物件：

  img = load_img("penguins.png")

- 使用Matplotlib套件來顯示這張圖片



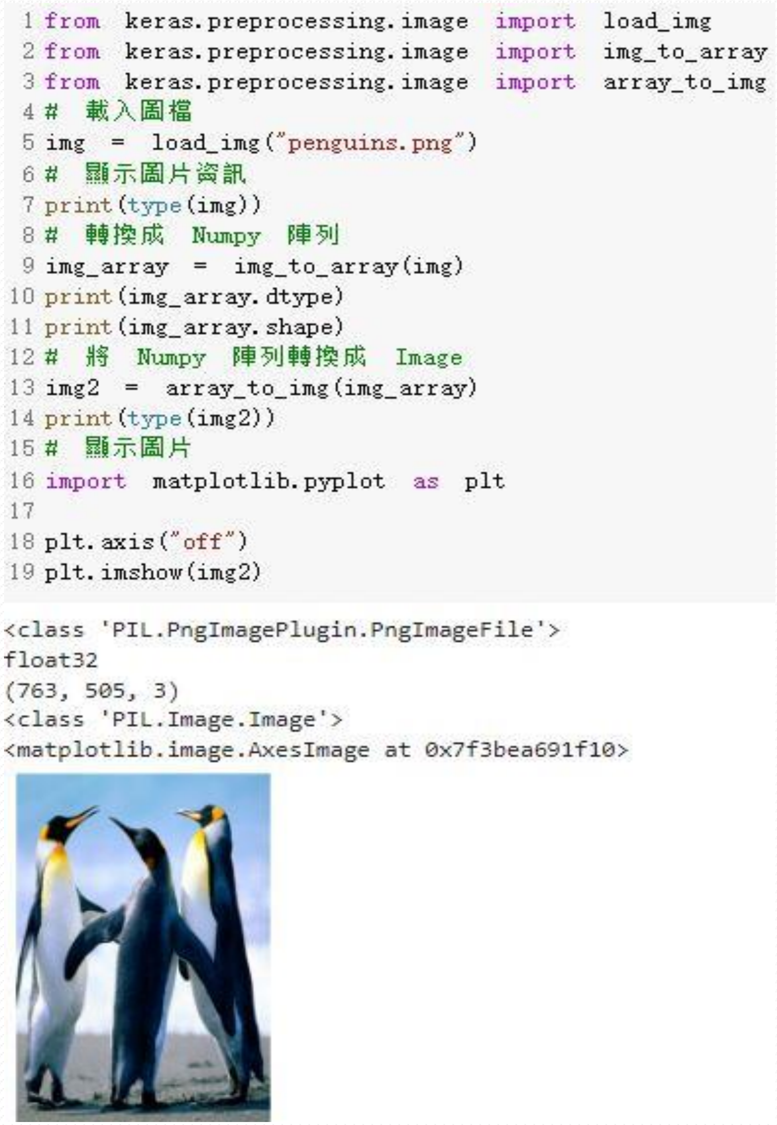
```
from  keras.preprocessing.image  import  load_img
#  載入圖檔
img  =  load_img("penguins.png")
#  顯示圖片資訊
print(type(img))
print(img.format)
print(img.mode)
print(img.size)
#  顯示圖片
import  matplotlib.pyplot  as  plt

plt.axis("off")
plt.imshow(img)
```

```
<class 'PIL.PngImagePlugin.PngImageFile'>
PNG
RGB
(505, 763)
<matplotlib.image.AxesImage at 0x7f3d78a4ab50>
```

將圖片轉換成NumPy陣列

- Python程式在呼叫load_img()函式載入圖檔後，就可以將圖片轉換成NumPy陣列

- 然後，我們可以呼叫array_to_img()函式，將NumPy陣列反過來轉換成Image物件的圖片

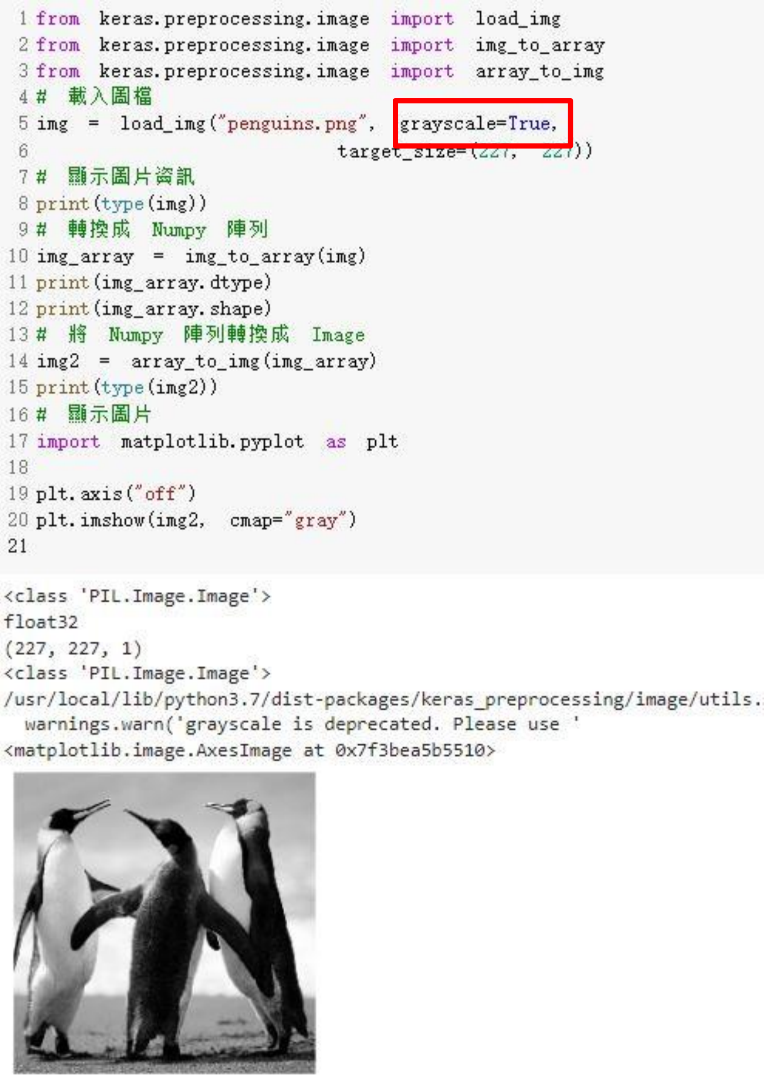


```
from  keras.preprocessing.image  import  load_img
from  keras.preprocessing.image  import  img_to_array
from  keras.preprocessing.image  import  array_to_img
#  載入圖檔
img  =  load_img("penguins.png")
#  顯示圖片資訊
print(type(img))
#  轉換成  Numpy  陣列
img_array  =  img_to_array(img)
print(img_array.dtype)
print(img_array.shape)
#  將  Numpy  陣列轉換成  Image
img2  =  array_to_img(img_array)
print(type(img2))
#  顯示圖片
import  matplotlib.pyplot  as  plt

plt.axis("off")
plt.imshow(img2)
```

```
<class 'PIL.PngImagePlugin.PngImageFile'>
float32
(763, 505, 3)
<class 'PIL.Image.Image'>
<matplotlib.image.AxesImage at 0x7f3bea691f10>
```

載入灰階圖片和調整圖片尺寸

- Keras的load_img()函式可以指定參數來將彩色圖片的圖檔自動載入成灰階圖片：

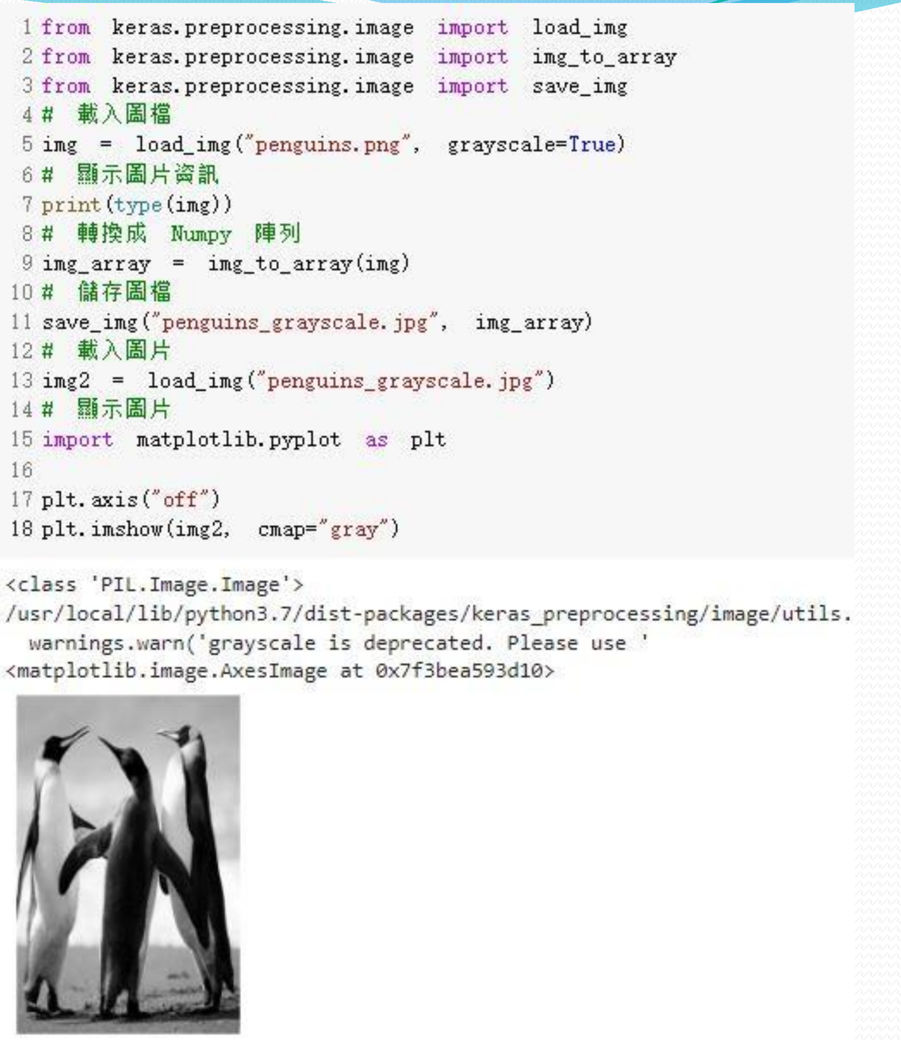


```
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import array_to_img
# 載入圖檔
img = load_img("penguins.png", grayscale=True,
                               target_size=(227, 227))
# 顯示圖片資訊
print(type(img))
# 轉換成 Numpy 陣列
img_array = img_to_array(img)
print(img_array.dtype)
print(img_array.shape)
# 將 Numpy 陣列轉換成 Image
img2 = array_to_img(img_array)
print(type(img2))
# 顯示圖片
import matplotlib.pyplot as plt

plt.axis("off")
plt.imshow(img2, cmap="gray")

```

```
<class 'PIL.Image.Image'>
float32
(227, 227, 1)
<class 'PIL.Image.Image'>
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/utils.
  warnings.warn('grayscale is deprecated. Please use '
<matplotlib.image.AxesImage at 0x7f3bea5b5510>
```

儲存圖片檔案

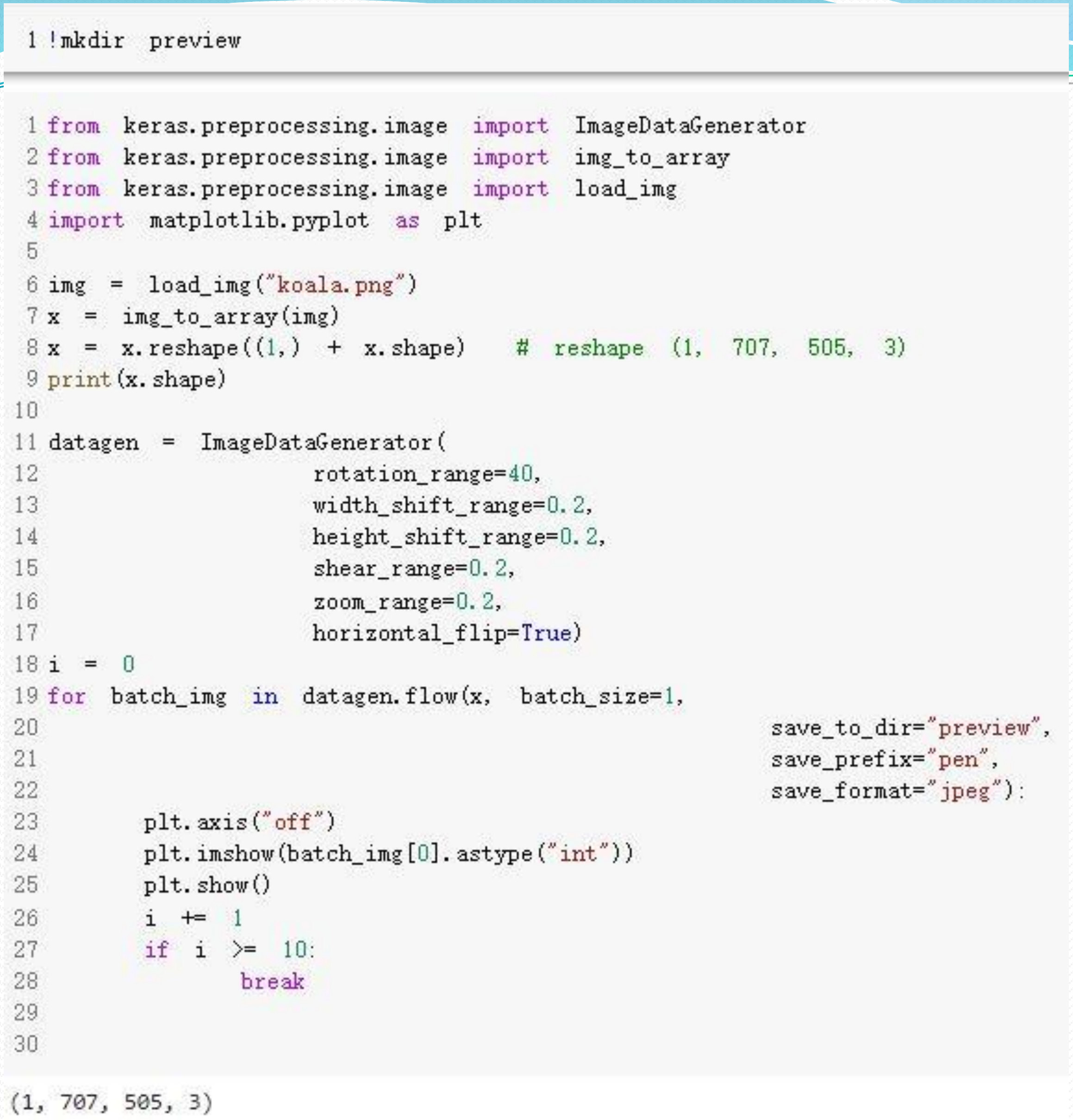
- Keras可以使用save_img()函式將圖片的NumPy陣列儲存成圖檔：



```python
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import save_img
# 載入圖檔
img = load_img("penguins.png", grayscale=True)
# 顯示圖片資訊
print(type(img))
# 轉換成 Numpy 陣列
img_array = img_to_array(img)
# 儲存圖檔
save_img("penguins_grayscale.jpg", img_array)
# 載入圖片
img2 = load_img("penguins_grayscale.jpg")
# 顯示圖片
import matplotlib.pyplot as plt

plt.axis("off")
plt.imshow(img2, cmap="gray")
```

```
<class 'PIL.Image.Image'>
/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/utils.
  warnings.warn('grayscale is deprecated. Please use '
<matplotlib.image.AxesImage at 0x7f3bea593d10>
```

資料增強



```
!mkdir  preview
```

```
from  keras.preprocessing.image  import  ImageDataGenerator
from  keras.preprocessing.image  import  img_to_array
from  keras.preprocessing.image  import  load_img
import  matplotlib.pyplot  as  plt

img  =  load_img("koala.png")
x  =  img_to_array(img)
x  =  x.reshape((1,)  +  x.shape)    #  reshape  (1,  707,  505,  3)
print(x.shape)

datagen  =  ImageDataGenerator(
                    rotation_range=40,
                    width_shift_range=0.2,
                    height_shift_range=0.2,
                    shear_range=0.2,
                    zoom_range=0.2,
                    horizontal_flip=True)
i  =  0
for  batch_img  in  datagen.flow(x,  batch_size=1,
                                                       save_to_dir="preview",
                                                       save_prefix="pen",
                                                       save_format="jpeg"):
       plt.axis("off")
       plt.imshow(batch_img[0].astype("int"))
       plt.show()
       i  +=  1
       if  i  >=  10:
              break


```

```
(1, 707, 505, 3)
```

要建立preview的文件夾，因為Generator要輸出到preview的文件夾裡

Keras的圖片增強API

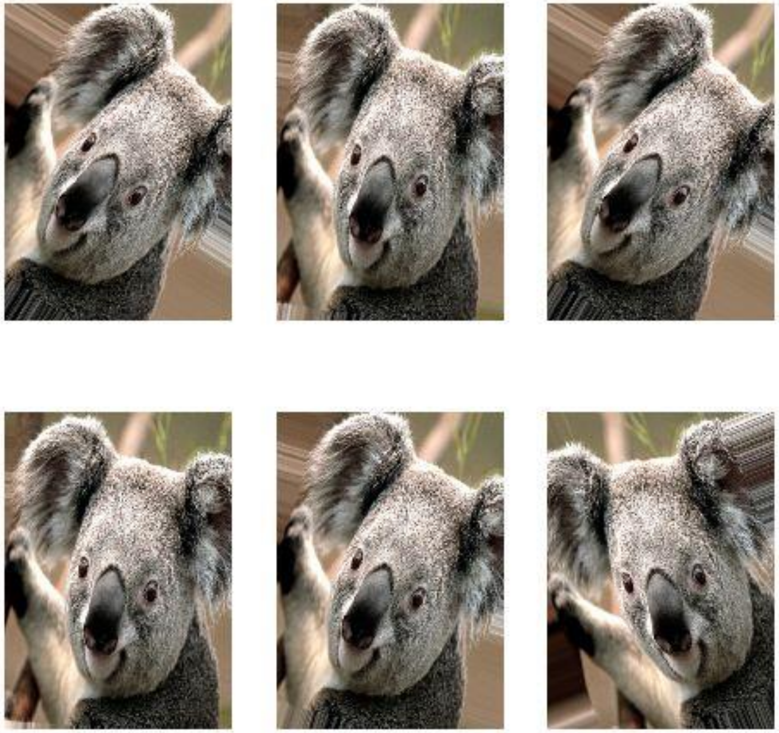
- 資料增強(Data Augmentation)在Keras是指「圖片增強」(Image Argumentation)，當訓練資料集的圖片數不足時，我們可以使用圖片增強技術來增加圖片的資料量。

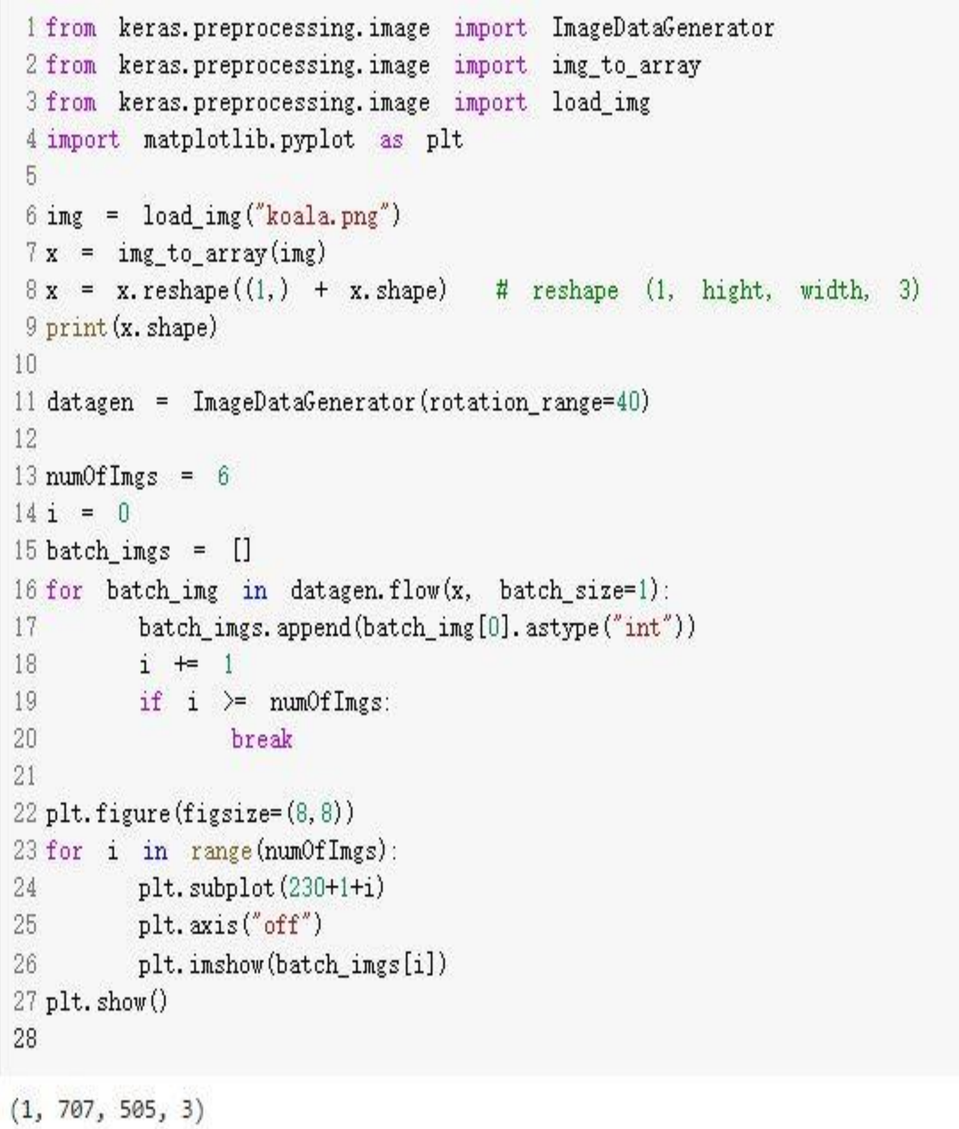
- 一張圖片只需經過旋轉、縮放、調整比例和翻轉等處理後，對於人類的眼睛來說，我們仍然可以輕鬆辨識出是同一張圖片，但是，對於深度學習模型來說，這些處理過的圖片就是一張全新圖片。

- 圖片增強是將資料集中的現有圖片，經過剪裁、旋轉、翻轉和縮放等操作來予以修改和變形，以便創造出更多圖片來增加訓練資料量，可以彌補訓練模型時資料量不足的問題。

隨機旋轉圖片

- 隨機旋轉(Random Rotations)可以隨機產生不同旋轉角度的增強圖片：



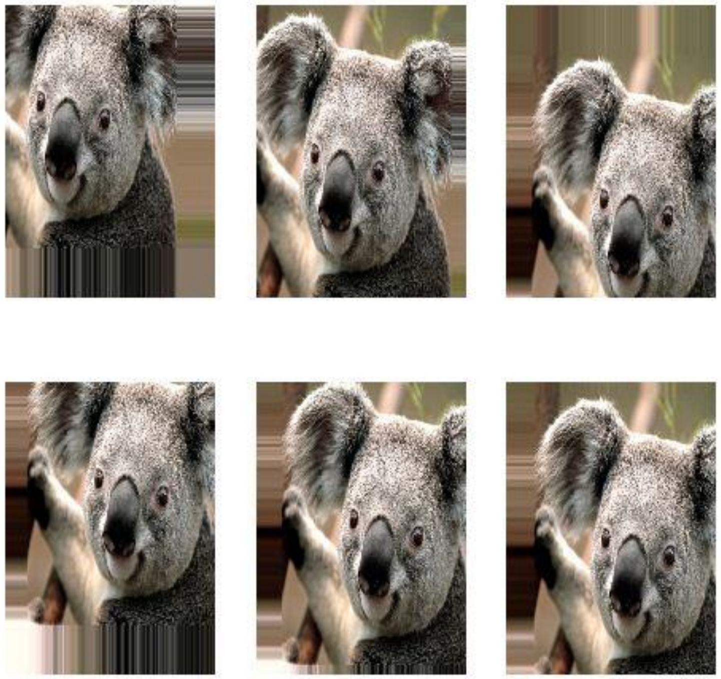


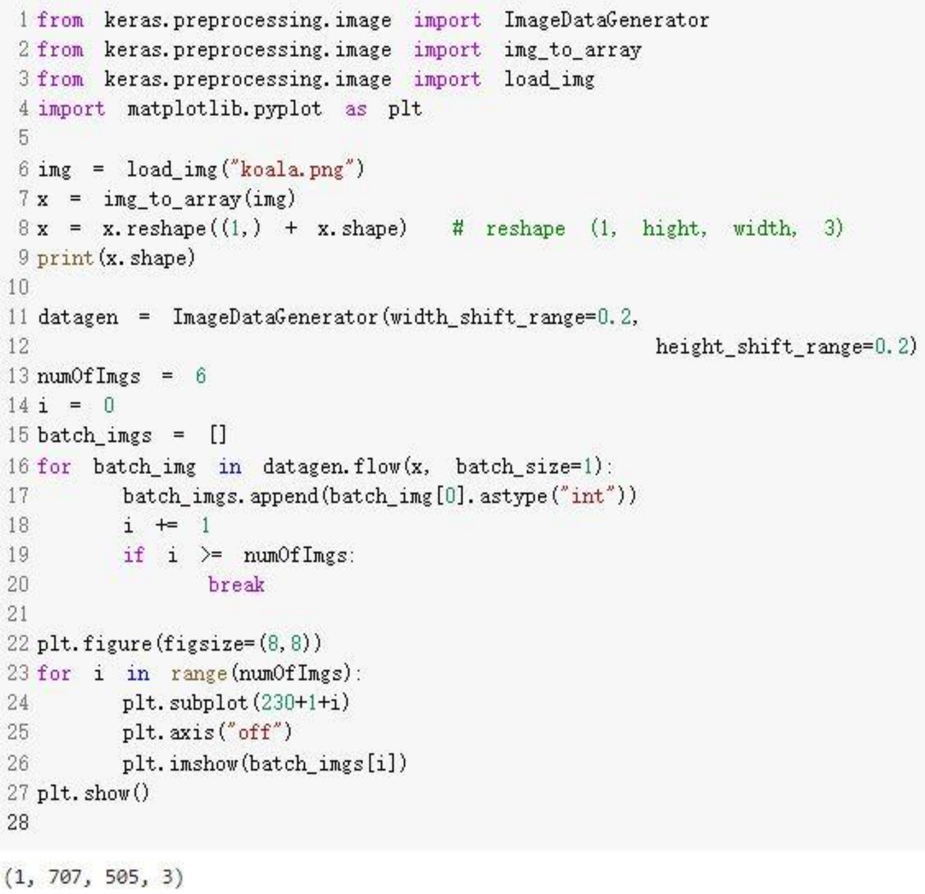
```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import matplotlib.pyplot as plt

img = load_img("koala.png")
x = img_to_array(img)
x = x.reshape((1,) + x.shape)   # reshape (1, hight, width, 3)
print(x.shape)

datagen = ImageDataGenerator(rotation_range=40)

numOfImgs = 6
i = 0
batch_imgs = []
for batch_img in datagen.flow(x, batch_size=1):
    batch_imgs.append(batch_img[0].astype("int"))
    i += 1
    if i >= numOfImgs:
        break

plt.figure(figsize=(8,8))
for i in range(numOfImgs):
    plt.subplot(230+1+i)
    plt.axis("off")
    plt.imshow(batch_imgs[i])
plt.show()

```

```
(1, 707, 505, 3)
```

隨機位移圖片

- 隨機位移(Random Shifts)可以隨機產生圖片中心點水平和垂直不同位移量的增強圖片。



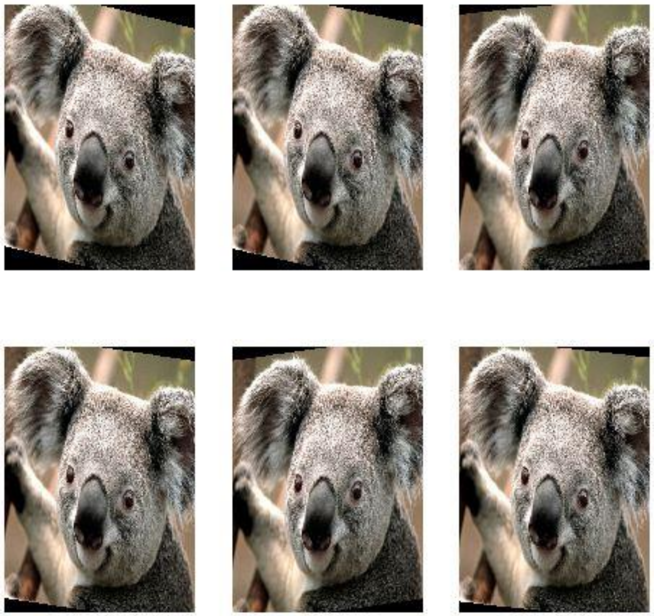


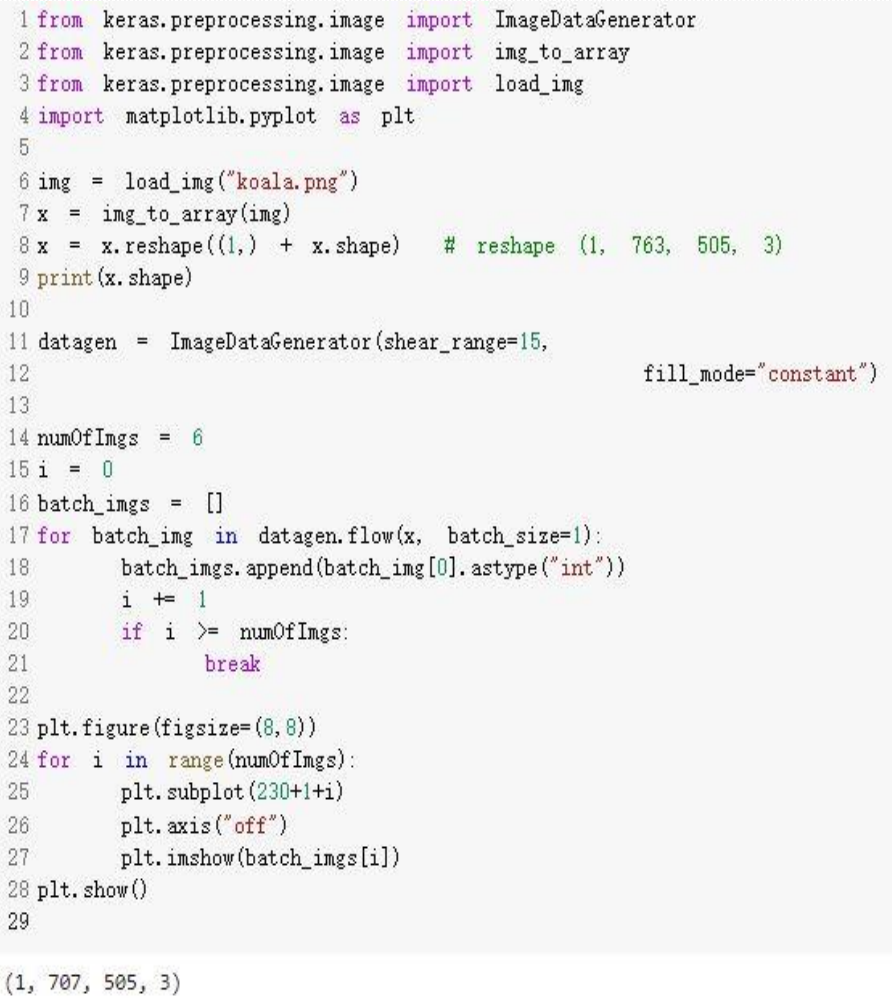
```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import matplotlib.pyplot as plt

img = load_img("koala.png")
x = img_to_array(img)
x = x.reshape((1,) + x.shape)   # reshape (1, hight, width, 3)
print(x.shape)

datagen = ImageDataGenerator(width_shift_range=0.2,
                                                   height_shift_range=0.2)
numOfImgs = 6
i = 0
batch_imgs = []
for batch_img in datagen.flow(x, batch_size=1):
        batch_imgs.append(batch_img[0].astype("int"))
        i += 1
        if i >= numOfImgs:
                break

plt.figure(figsize=(8,8))
for i in range(numOfImgs):
        plt.subplot(230+1+i)
        plt.axis("off")
        plt.imshow(batch_imgs[i])
plt.show()

```

```
(1, 707, 505, 3)
```

隨機推移變換圖片

- 隨機推移變換(Random Shears)是在垂直軸不動來隨機推移變形產生增強圖片，在ImageDataGenerator物件是使用shear_range參數



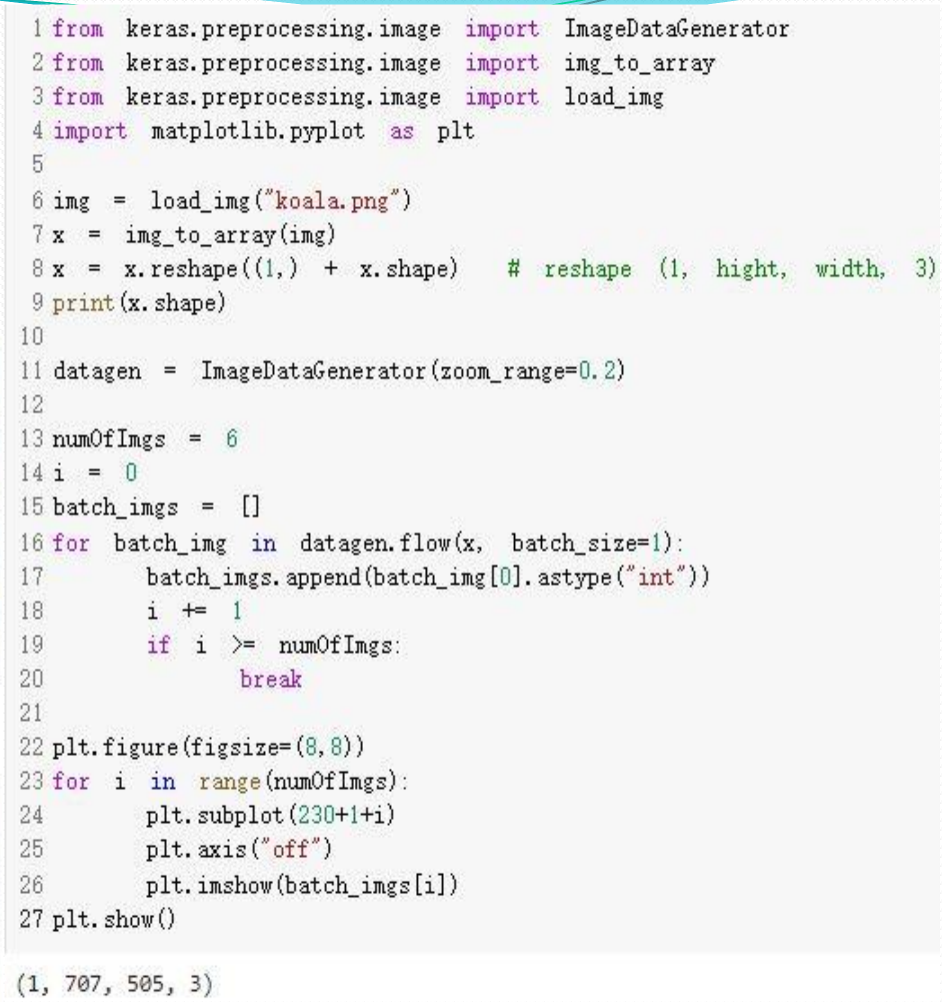


```python
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import matplotlib.pyplot as plt

img = load_img("koala.png")
x = img_to_array(img)
x = x.reshape((1,) + x.shape)   # reshape (1, 763, 505, 3)
print(x.shape)

datagen = ImageDataGenerator(shear_range=15,
                             fill_mode="constant")


numOfImgs = 6
i = 0
batch_imgs = []
for batch_img in datagen.flow(x, batch_size=1):
    batch_imgs.append(batch_img[0].astype("int"))
    i += 1
    if i >= numOfImgs:
        break

plt.figure(figsize=(8,8))
for i in range(numOfImgs):
    plt.subplot(230+1+i)
    plt.axis("off")
    plt.imshow(batch_imgs[i])
plt.show()

```

```
(1, 707, 505, 3)
```

隨機縮放圖片

- 隨機縮放（Random Zooms）可以隨機產生不同圖片縮放的增強圖片。

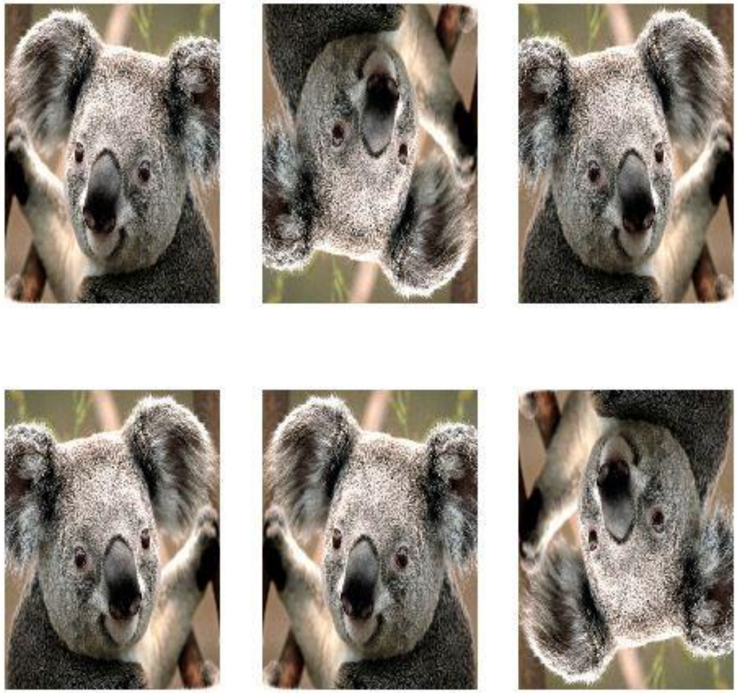


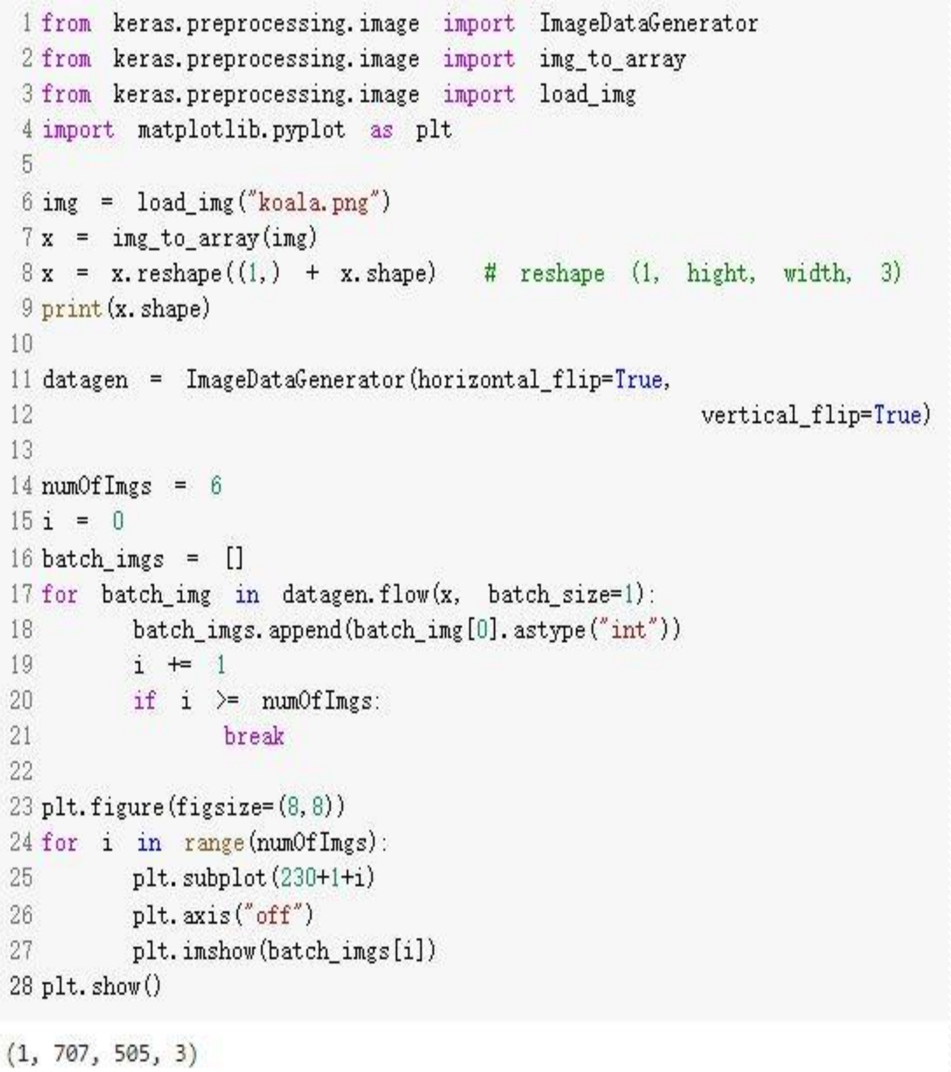
```python
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import matplotlib.pyplot as plt

img = load_img("koala.png")
x = img_to_array(img)
x = x.reshape((1,) + x.shape)   # reshape (1, hight, width, 3)
print(x.shape)

datagen = ImageDataGenerator(zoom_range=0.2)

numOfImgs = 6
i = 0
batch_imgs = []
for batch_img in datagen.flow(x, batch_size=1):
    batch_imgs.append(batch_img[0].astype("int"))
    i += 1
    if i >= numOfImgs:
        break

plt.figure(figsize=(8,8))
for i in range(numOfImgs):
    plt.subplot(230+1+i)
    plt.axis("off")
    plt.imshow(batch_imgs[i])
plt.show()
```

```
(1, 707, 505, 3)
```

隨機翻轉圖片

- 隨機翻轉(Random Flips)可以隨機產生圖片水平和垂直翻轉的增強圖片

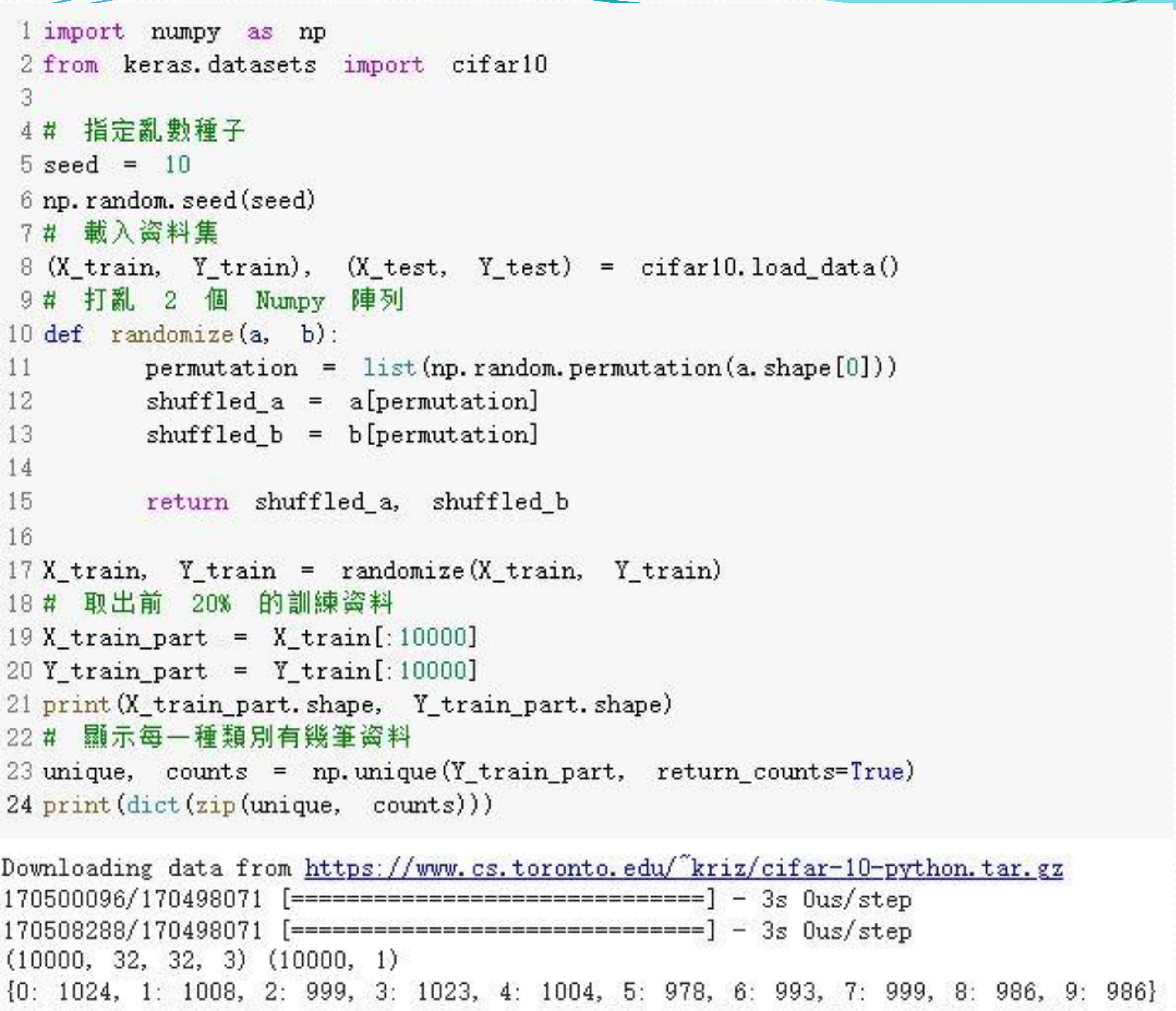




```python
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import matplotlib.pyplot as plt

img = load_img("koala.png")
x = img_to_array(img)
x = x.reshape((1,) + x.shape)   # reshape (1, hight, width, 3)
print(x.shape)

datagen = ImageDataGenerator(horizontal_flip=True,
                             vertical_flip=True)

numOfImgs = 6
i = 0
batch_imgs = []
for batch_img in datagen.flow(x, batch_size=1):
    batch_imgs.append(batch_img[0].astype("int"))
    i += 1
    if i >= numOfImgs:
        break

plt.figure(figsize=(8,8))
for i in range(numOfImgs):
    plt.subplot(230+1+i)
    plt.axis("off")
    plt.imshow(batch_imgs[i])
plt.show()
```

```
(1, 707, 505, 3)
```

Cifar-10資料集的
小資料量圖片分類



```
import numpy as np
from keras.datasets import cifar10

# 指定亂數種子
seed = 10
np.random.seed(seed)
# 載入資料集
(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()
# 打亂 2 個 Numpy 陣列
def randomize(a, b):
        permutation = list(np.random.permutation(a.shape[0]))
        shuffled_a = a[permutation]
        shuffled_b = b[permutation]

        return shuffled_a, shuffled_b

X_train, Y_train = randomize(X_train, Y_train)
# 取出前 20% 的訓練資料
X_train_part = X_train[:10000]
Y_train_part = Y_train[:10000]
print(X_train_part.shape, Y_train_part.shape)
# 顯示每一種類別有幾筆資料
unique, counts = np.unique(Y_train_part, return_counts=True)
print(dict(zip(unique, counts)))
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 3s 0us/step
170508288/170498071 [==============================] - 3s 0us/step
(10000, 32, 32, 3) (10000, 1)
{0: 1024, 1: 1008, 2: 999, 3: 1023, 4: 1004, 5: 978, 6: 993, 7: 999, 8: 986, 9: 986}
```

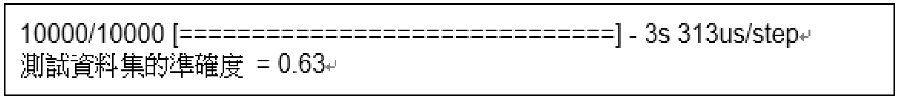
取出Cifar-10資料集的部分訓練資料

- 前面章節已經說明過Cifar-10資料集的圖片分類，10個分類中，每一類有6,000張圖片，分成50,000張訓練資料集和10,000張測試資料集。我們準備只取出10,000張圖片來訓練圖片分類模型。
- 本Python程式只準備取出前10,000張圖片，為了更隨機，所以建立打亂資料的randomize()函式，可以打亂參數的2 個 NumPy陣列，接著，就呼叫randomize()函式來打亂訓練資料集：

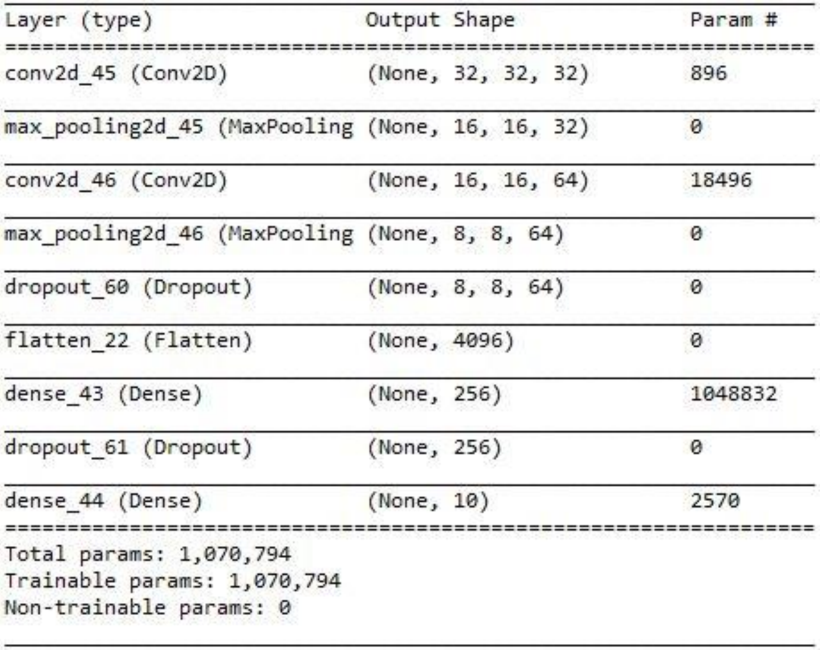
```
X_train, Y_train = randomize(X_train, Y_train)
```

沒有圖片增強的小資料量圖片分類CNN模型

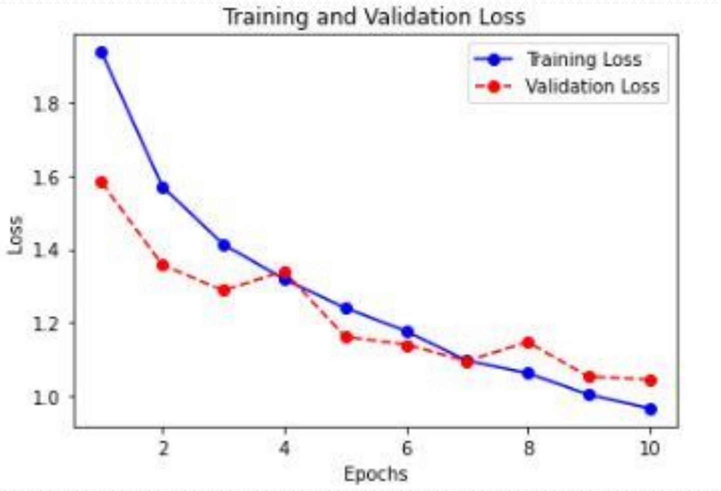
- 本Python程式是使用第13-5-1節分割出的前10,000張圖片來訓練圖片分類模型，這一節程式並沒有使用Keras圖片增強API。在CNN模型是使用2組卷積和池化層。



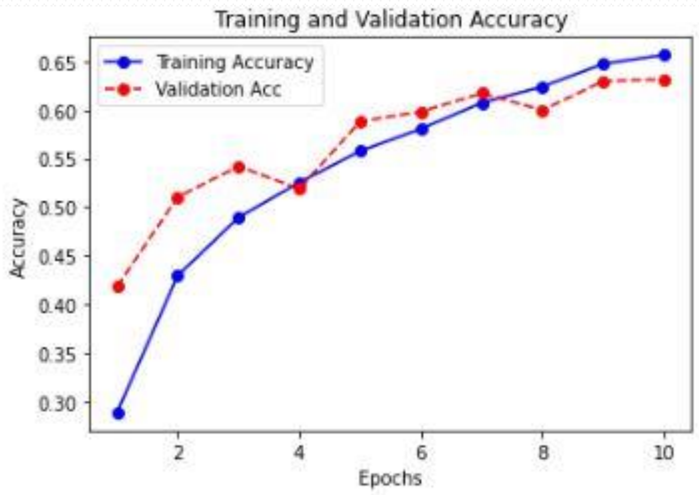
```
10000/10000 [==============================] - 3s 313us/step↵
測試資料集的準確度 = 0.63↵
```



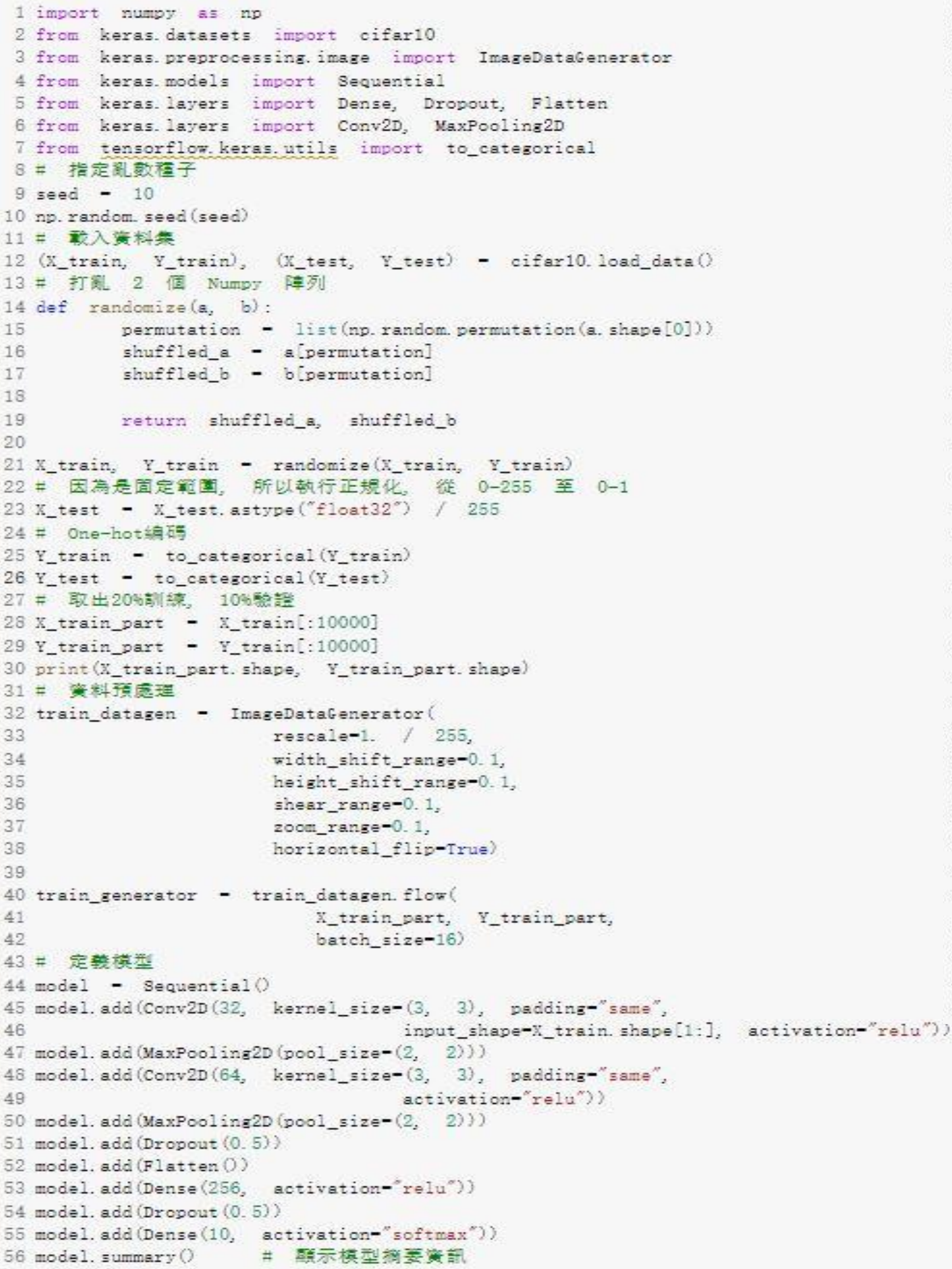
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_45 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_45 (MaxPooling (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_46 (Conv2D)           (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_46 (MaxPooling (None, 8, 8, 64)          0
_________________________________________________________________
dropout_60 (Dropout)         (None, 8, 8, 64)          0
_________________________________________________________________
flatten_22 (Flatten)         (None, 4096)              0
_________________________________________________________________
dense_43 (Dense)             (None, 256)               1048832
_________________________________________________________________
dropout_61 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_44 (Dense)             (None, 10)                2570
=================================================================
Total params: 1,070,794
Trainable params: 1,070,794
Non-trainable params: 0
_________________________________________________________________
```



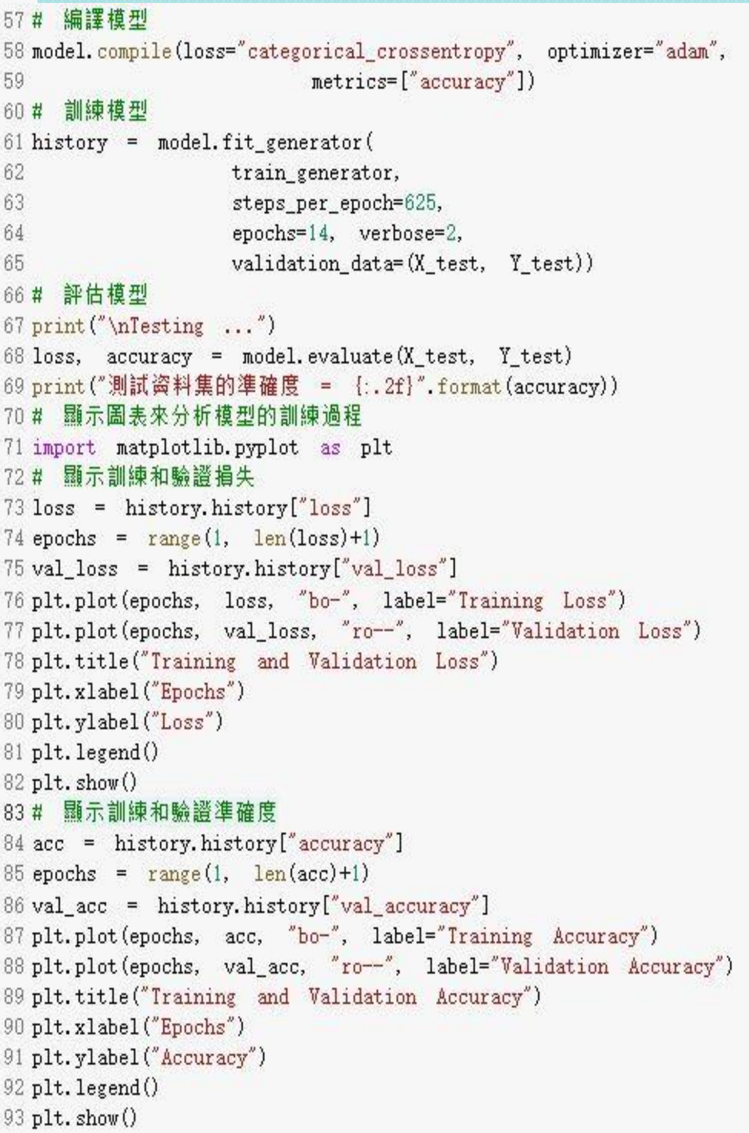
訓練和驗證損失的趨勢圖表



訓練和驗證準確度的趨勢圖表



```python
import numpy as np
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.utils import to_categorical
# 指定亂數種子
seed = 10
np.random.seed(seed)
# 載入資料集
(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()
# 打亂 2 個 Numpy 陣列
def randomize(a, b):
        permutation = list(np.random.permutation(a.shape[0]))
        shuffled_a = a[permutation]
        shuffled_b = b[permutation]

        return shuffled_a, shuffled_b

X_train, Y_train = randomize(X_train, Y_train)
# 因為是固定範圍, 所以執行正規化, 從 0-255 至 0-1
X_test = X_test.astype("float32") / 255
# One-hot編碼
Y_train = to_categorical(Y_train)
Y_test = to_categorical(Y_test)
# 取出20%訓練, 10%驗證
X_train_part = X_train[:10000]
Y_train_part = Y_train[:10000]
print(X_train_part.shape, Y_train_part.shape)
# 資料預處理
train_datagen = ImageDataGenerator(
                    rescale=1. / 255,
                    width_shift_range=0.1,
                    height_shift_range=0.1,
                    shear_range=0.1,
                    zoom_range=0.1,
                    horizontal_flip=True)

train_generator = train_datagen.flow(
                        X_train_part, Y_train_part,
                        batch_size=16)
# 定義模型
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), padding="same",
                        input_shape=X_train.shape[1:], activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), padding="same",
                        activation="relu"))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation="softmax"))
model.summary()     # 顯示模型摘要資訊
```

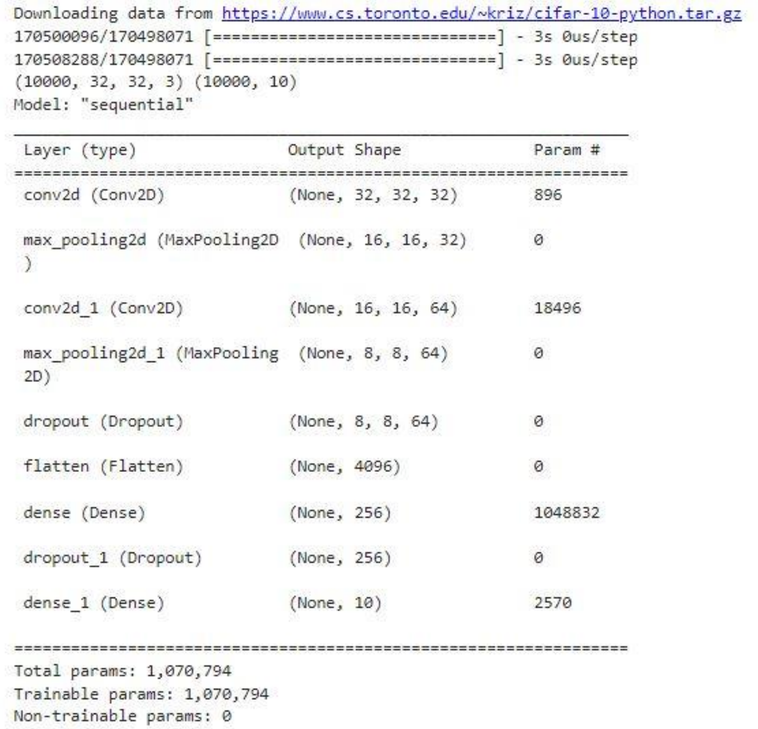


```python
# 編譯模型
model.compile(loss="categorical_crossentropy", optimizer="adam",
                        metrics=["accuracy"])
# 訓練模型
history = model.fit_generator(
                train_generator,
                steps_per_epoch=625,
                epochs=14, verbose=2,
                validation_data=(X_test, Y_test))
# 評估模型
print("\nTesting ...")
loss, accuracy = model.evaluate(X_test, Y_test)
print("測試資料集的準確度 = {:.2f}".format(accuracy))
# 顯示圖表來分析模型的訓練過程
import matplotlib.pyplot as plt
# 顯示訓練和驗證損失
loss = history.history["loss"]
epochs = range(1, len(loss)+1)
val_loss = history.history["val_loss"]
plt.plot(epochs, loss, "bo-", label="Training Loss")
plt.plot(epochs, val_loss, "ro--", label="Validation Loss")
plt.title("Training and Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
# 顯示訓練和驗證準確度
acc = history.history["accuracy"]
epochs = range(1, len(acc)+1)
val_acc = history.history["val_accuracy"]
plt.plot(epochs, acc, "bo-", label="Training Accuracy")
plt.plot(epochs, val_acc, "ro--", label="Validation Accuracy")
plt.title("Training and Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

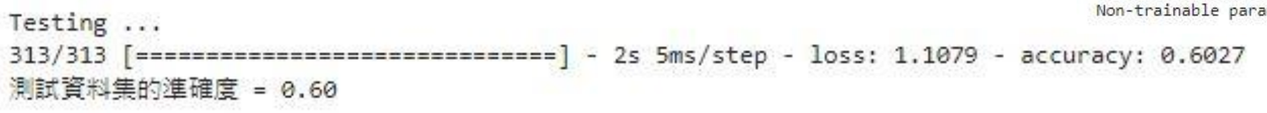
前面有設定batch size等於steps_per_epoch=625 (10000/16)

使用圖片增強的小資料量圖片分類使用圖片增強API

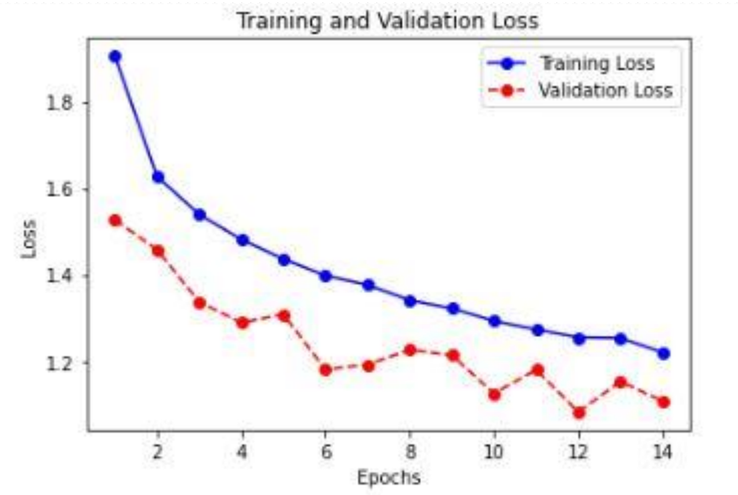
- 本Python程式使用和前一節相同的CNN模型，只是使用圖片增強API來增加訓練圖片的資料量。
- 請注意！因為需要產生增強圖片，訓練模型不是使用fit()函式，而是呼叫fit_generator()函式。

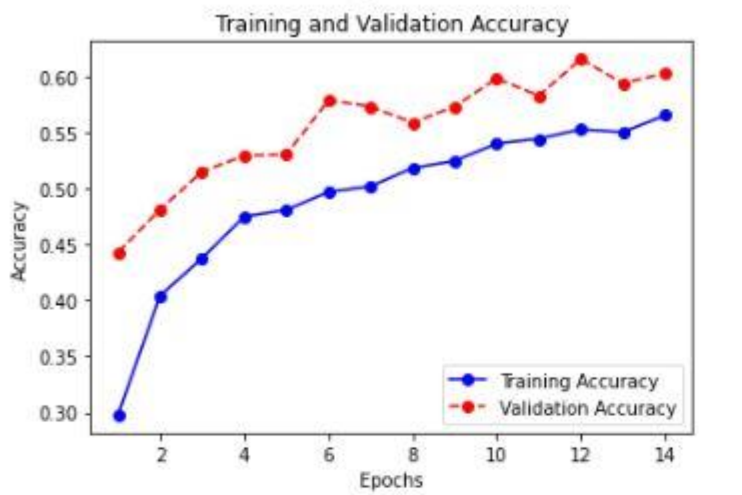


```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 3s 0us/step
170508288/170498071 [==============================] - 3s 0us/step
(10000, 32, 32, 3) (10000, 10)
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 16, 16, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout (Dropout)           (None, 8, 8, 64)          0

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 256)               1048832

 dropout_1 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 1,070,794
Trainable params: 1,070,794
Non-trainable params: 0
```



```
Testing ...
313/313 [==============================] - 2s 5ms/step - loss: 1.1079 - accuracy: 0.6027
測試資料集的準確度 = 0.60
```





End !