卷積神經網路 Convolutional Neural Network
&
電腦視覺 Computer Vision
Part3

林彥宇&教研處

「版權聲明頁」

本投影片已經獲得作者授權台灣人工智慧學校得以使用於教學用途，如需取得重製權以及公開傳輸權需要透過台灣人工智慧學校取得著作人同意；如果需要修改本投影片著作，則需要取得改作權；另外，如果有需要以光碟或紙本等實體的方式傳播，則需要取得人工智慧學校散佈權。

–台灣人工智慧學校

課程內容

本日課程:

1. CNN實戰 in Keras
2. Generator(Data Augmentation)

延伸閱讀 (Optional):

1. What does CNN learn?

本次課程結束後你 (妳) 應該會什麼?

- **軟實力**
  - 理解data augmentation技法
  - 了解CNN實際學到的內容是什麼(optional)
- **硬底子**
  - 使用 Keras 寫出基本的 CNN
  - 用Simpson dataset練習CNN與data augmentation
  - 學會手刻generator或使用Keras內建的generator

CNN in Keras

[實戰演練]CNN in Keras:

CNN 的訓練技巧

- Data augmentation

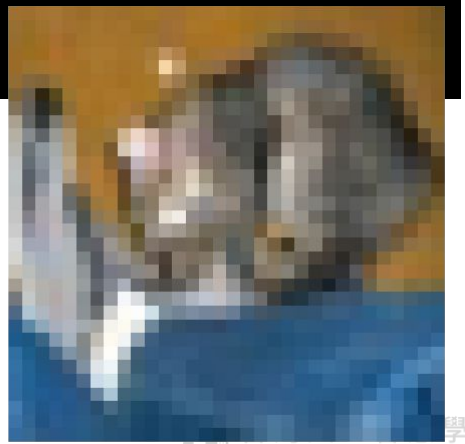
How to Load CIFAR-10 Dataset by Keras

- This reading function is provided from the Keras

```
# this function is provided from the official site
from keras.datasets import cifar10

# read train/test data
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# check the data shape
print("x_train shape:", x_train.shape)
print("numbers of training smaples:", x_train.shape[0])
print("numbers of testing smaples:", x_test.shape[0])
```

Show the images

```
import matplotlib.pyplot as plt
%matplotlib inline

# show the first image of training data
plt.imshow(x_train[0])

# show the first image of testing data
plt.imshow(x_test[0])
```



Building CNN model

32 個 3x3 filters

'valid' : without padding
'same': perform padding
default value is zero

```
'''CNN model'''
model = Sequential()
model.add(                                                    CNN
        Convolution2D(32, 3, 3, padding='same',
                      input_shape=x_train[0].shape)
        )
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))                                         DNN
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10))
model.add(Activation('softmax'))
```

Model Compilation

```
'''setting optimizer'''

# define the optimizer
optimizer = Adam()

# Let's compile our model
model.compile(loss='categorical_crossentropy', optimizer= optimizer,
metrics=['accuracy'])
```

Number of Parameters of Each Layer

- Only one function

```
# check parameters of every layers
model.summary()
```

- Total parameters

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
convolution2d_5 (Convolution2D)  (None, 32, 32, 32)    896
_________________________________________________________________
convolution2d_6 (Convolution2D)  (None, 30, 30, 32)    9248
_________________________________________________________________
maxpooling2d_3 (MaxPooling2D)    (None, 15, 15, 32)    0
_________________________________________________________________
dropout_5 (Dropout)          (None, 15, 15, 32)    0
_________________________________________________________________
flatten_3 (Flatten)          (None, 7200)          0
_________________________________________________________________
dense_5 (Dense)              (None, 512)           3686912
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)           0
_________________________________________________________________
dense_6 (Dense)              (None, 10)            5130
=================================================================
Total params: 3702186
```

Number of Parameters of Each Layer

Layer (type)	Output Shape	Param #
convolution2d_1 (Convolution2D)	(None, 32, 32, 32)	896
activation_1 (Activation)	(None, 32, 32, 32)	0
convolution2d_2 (Convolution2D)	(None, 30, 30, 32)	9248
activation_2 (Activation)	(None, 30, 30, 32)	0
maxpooling2d_1 (MaxPooling2D)	(None, 15, 15, 32)	0
dropout_1 (Dropout)	(None, 15, 15, 32)	0
flatten_1 (Flatten)	(None, 7200)	0
dense_1 (Dense)	(None, 512)	3686912
activation_3 (Activation)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130
activation_4 (Activation)	(None, 10)	0

Total params: 3702186

3*3*3*32 + 32

32*3*3*32 + 32

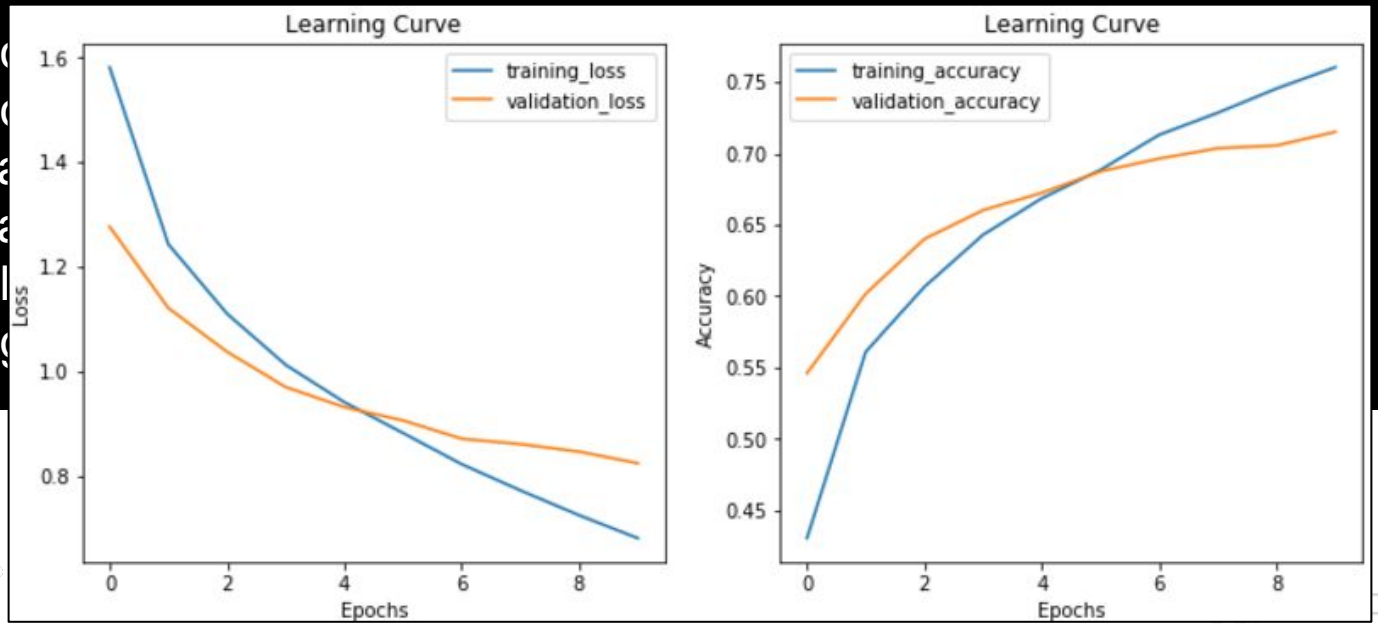
7200*512 + 512

Let’s Start Training

```
# define batch size and # of epoch
batch_size = 32 or 64 or 128
epoch = 10

# [2] validation data comes from testing data
model_history = model.fit(x_train, y_train, batch_size=batch_size,
epochs=epoch, shuffle=True,
validation_data=(x_test, y_test))
```

Plot Training History

```
# get learning loss from model_history.history
training_loss = model_history.history['loss']
testing_loss = model_history.history['val_loss']

plt.plo
plt.plo
plt.xla
plt.yla
plt.titl
plt.leg
```



Model Saving and Prediction

- Saving/loading the whole CNN model

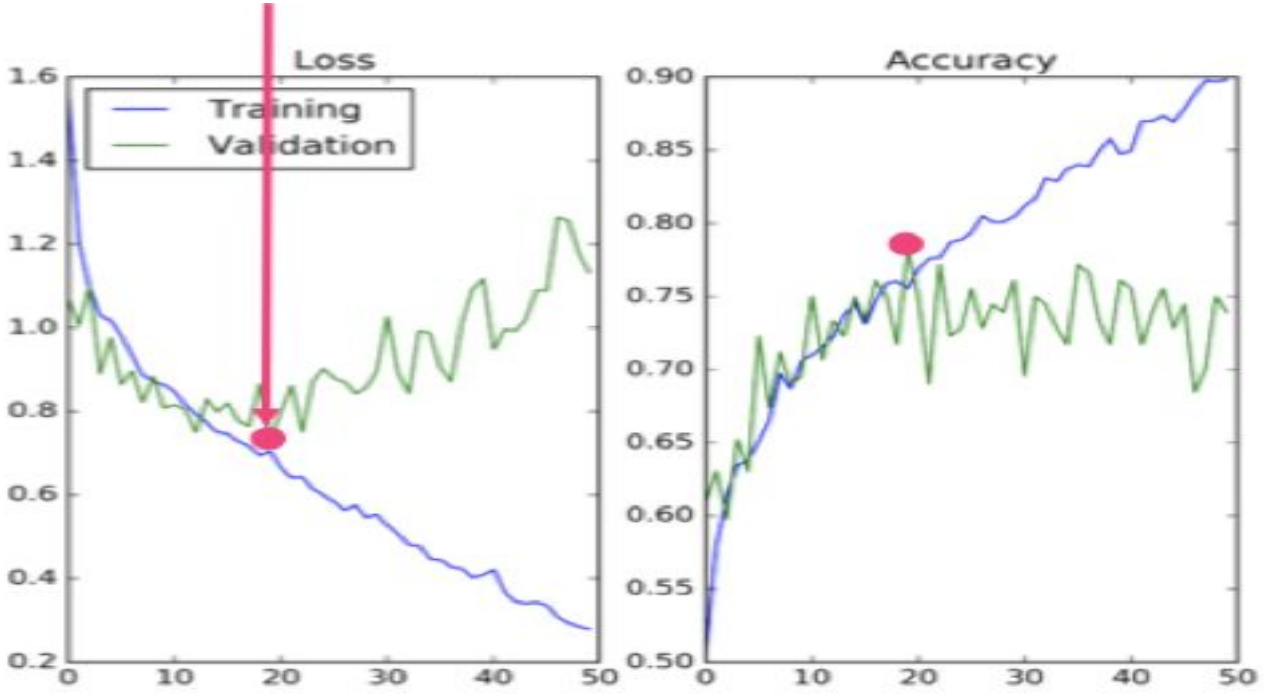
```
'''saving model'''
from keras.models import load_model
model.save('cifar10.h5')
del model

'''loading model'''
model = load_model('cifar10.h5')
```

- Predicting the classes with new image samples

```
'''prediction'''
y_pred = model.predict_classes(x_test, batch_size, verbose=0)
```

CNN 的訓練技巧

- Early Stopping



CNN 的訓練技巧

- Data augmentation

Data augmentation in Keras

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(
                rotation_range=30,
                width_shift_range=0.1,
                height_shift_range=0.1,
                horizontal_flip=True,
                     vertical_flip=False)
model.fit_generator(datagen.flow(x_train, y_train))
```

Earlystop in Keras

```
from keras.callbacks  import EarlyStopping
earlystop = EarlyStopping(
                        monitor=”val_loss”,
                      patience=5)
model.fit_generator(datagen.flow(x_train, y_train),
                      callbacks=earlystop)
```

Data Augmentation

What is data augmentation？

在資料有限的情況下，以數據處理的方式來增加大量資料。以圖片的資料增強來說，對圖片進行翻轉、旋轉、放大縮小、平移、明暗度或色調的改變，都是資料增強的一種手法。

Why data augmentation？

通常來說，無論是機器學習或者是深度學習，都會希望是有效的訓練資料是越多越好，能夠讓所訓練的模型更強健。而在實務上一個被拍到的物體或是動物，實際上的認知並不會因為在照片上的大小、位置、角度各種狀況，在學習資料不足的情況下就視為不同的東西，而透過資料增強的方式，能夠有效的解決。

How to do data augmentation？

最直接土法煉鋼的方式，就是透過Open Computer Vision(OpenCV)的API，對圖片進行放大縮小翻轉平移，甚至模糊處理，這些方式都能夠過OpenCV實現。

而在Keras之下有提供更為方便資料增量的API (ImageDataGenerator)。

Keras-ImageDataGenerator

所需套件

```
from keras.preprocessing.image import ImageDataGenerator, load_img,
img_to_array

from glob import glob
import matplotlib.pyplot as plt
import cv2
import numpy as np
```

常用

```
datagen = ImageDataGenerator(
                    rotation_range=45,              #旋轉
                    width_shift_range=0.1,                  #寬度縮放
                    height_shift_range=0.1,                 #高度縮放
                    rescale=1./255,                                 #重新縮放
                    shear_range=0.4,                            #斷開位移(
                    zoom_range=0.3,                             #局部放大
                    horizontal_flip=True,                   #平移
                    channel_shift_range=50,             #色調變形
                    fill_mode='nearest') #當圖片扭曲時所產生的空白邊界進行填充。
```

更詳細請參考

https://keras.io/preprocessing/image/

範例

```
for batch in datagen.flow(x,batch_size=1,save_to_dir='./datagen', #產生的資料保存的位置
        save_prefix='cat',save_format='png'):

    i +=1
    plt.subplot(5,4,1-1+ i)
    plt.axis("off")
    augImg = batch[0]
    plt.imshow(augImg)
    if i > 7 :
        break
```

batch_size 為一次所產生的圖片數量，若最後沒設定break則會不停的產生圖片。

常見疑問

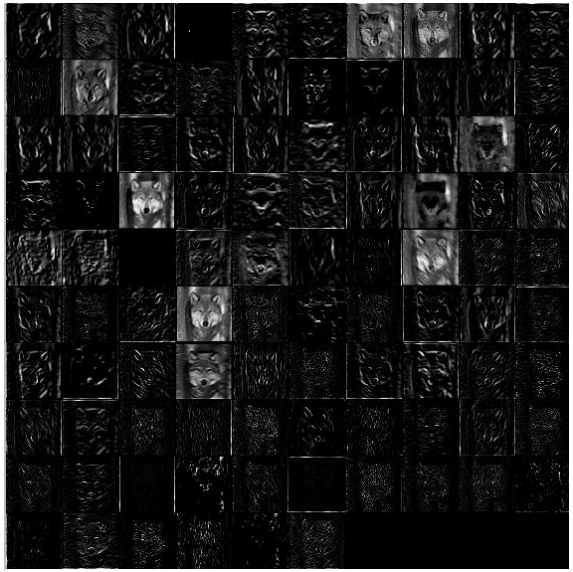
如果同學們有使用過CNN課程part1的範例，會看到下列這兩部分的程式碼

```
datagen = ImageDataGenerator(
    rotation_range=30,  # randomly rotate images in the range (degrees, 0 to 180)
    width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
    horizontal_flip=True,  # randomly flip images
    vertical_flip=False)  # randomly flip images

model_history = model.fit_generator(datagen.flow(x_train, y_train,
                                    batch_size=batch_size),
                epochs=epochs,
                validation_data=(x_test, y_test),
                workers=4,
                callbacks=[earlystop])
```

而到底做了什麼呢？
簡單來說，就是每個Epoch從train data中隨機抽出不同的data並進行資料增強後一次吐出batch_size數量的圖片。
而不是將data增加數倍數量放在那邊再給神經網路學習，這邊很容易造成同學們誤會。

CNN 的架構建議

- 建議使用 3x3 的 filter size
- filter 數量應該要越來越多 64 -> 128 -> 256
- 盡量別做太多的 Maxpooling 以免丟失過多訊息
- 使用 Data-augmentation 增加資料量
- 使用 Earlystop 避免 Overfitting
- 若嚴重 Overfitting 可嘗試 dropout, L2 regularization

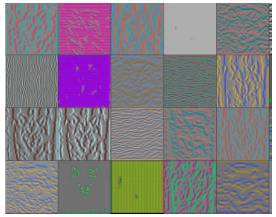


延伸閱讀-1

What does CNN learn?

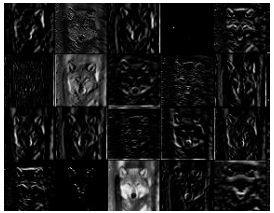
CNN到底看到了什麼，又學了什麼？

- 我們在訓練完CNN後得到的feature map到底長什麼樣子？ ->



- 餵圖片給訓練完之後的CNN，經過第一層的feature map後原圖變成什麼樣子？

  ->



[實作範例](Optional)

- 在part3/03_feature_map_optional 資料夾中
- 檔案都已經下載好，但仍需在terminal中執行以下指令，才能呼叫模組使用。
  1. **cd 你複製的資料夾位置**
  2. **cd feature_map_optional**
  3. **python setup.py install**
- 接下來在example中就有範例囉，若有興趣可以在以下的Reference中找到更多資訊。
- Reference: SourceCode來源與介紹