



深度學習TensorFlow實務

CNN 圖片分類

Lab3

-TA-

李偉弘

廖宜健

林佑昌

蔡明諺

彭冠偉

1. 卷積是什麼

卷積(Convolution)

- Convolution 是一種函數的定義
- 透過兩個函數 f 和 g 生成第三個函數的一種數學運算

2. 卷積核(Convolution Kernel)

Convolution Kernel

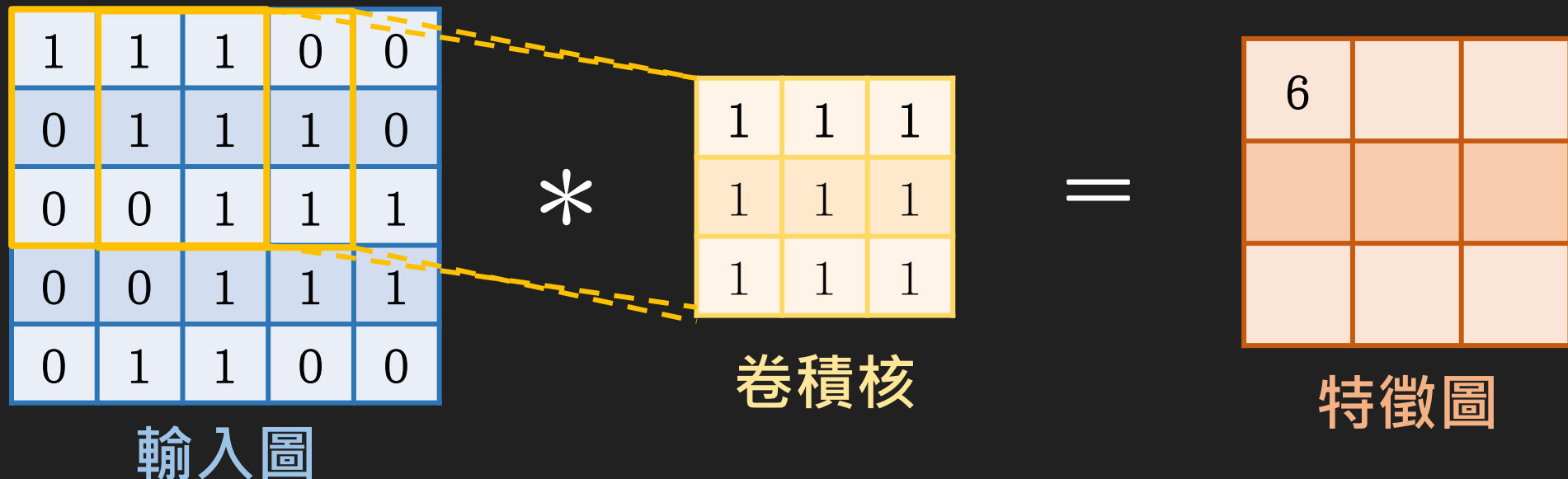
- 卷積核又可以稱為Filter、Window，我們可以理解它是在影像中滑動去提取特徵
- 卷積核的表達式為： $f(x) = xw + b$
- 我們設計一個特別簡單的卷積核：

$$w = [1, 1, 1, 1, 1, 1, 1, 1, 1]$$

$$b = [0]$$

1	1	1
1	1	1
1	1	1

Convolution 運算



- 假設圖片有 5×5 一共 25 個像素點，像素點只有 1 和 0
- 圖中黃色 3×3 部分 w 由 9 個 1 構成，從左上到右下的這 9 個點作為 x 向量，每個與 w 相乘完成內積操作並與 b 相加

$$f(x) = 1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 1 \times 1 + 1 \times 1 + 1 \times 0 + 1 \times 0 + 1 \times 1 = 6$$

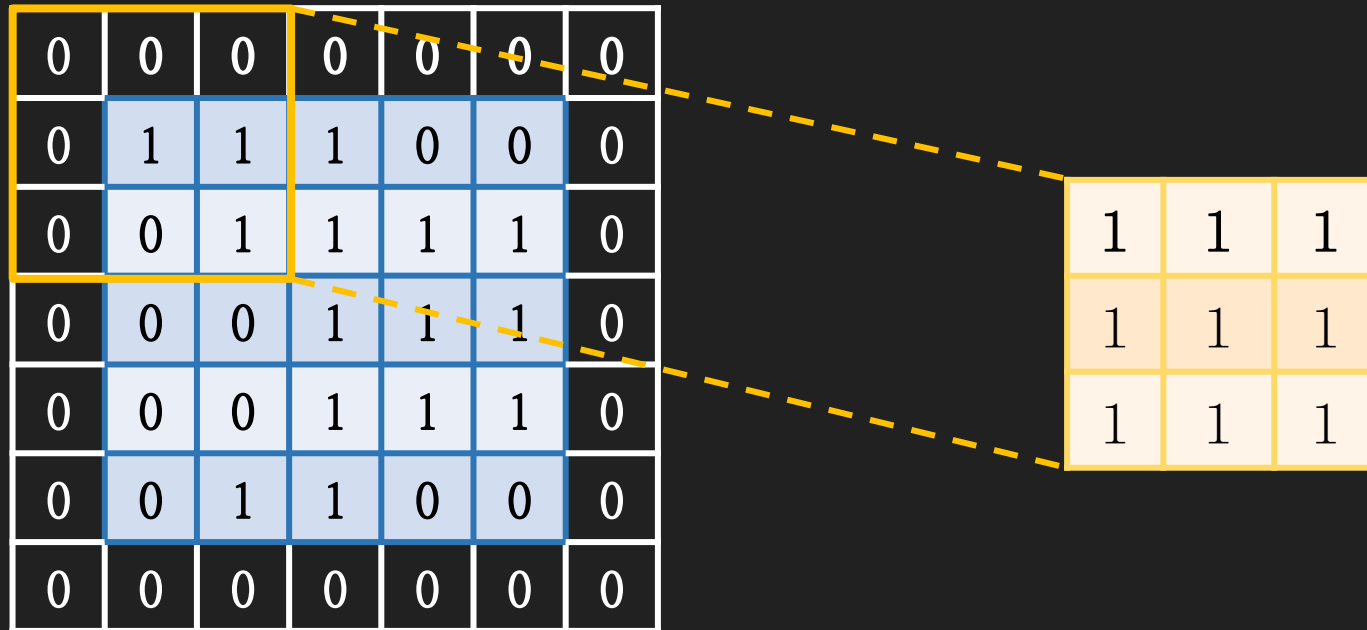
3. 卷積層其他參數

卷積層其他參數

- 在卷積核前面輸入的這一層資料向量進行掃描時有幾個參數需要注意，分別為：
 - 填充 (Padding)
 - 步幅 (Stride)

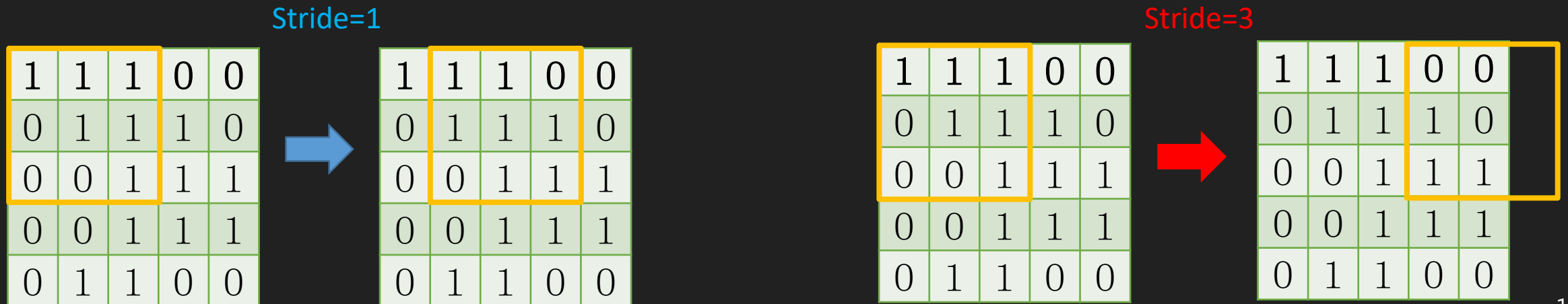
卷積層其他參數

- Padding是指用多少個像素單位來填充輸入影像(向量)的邊界，通常都是填充0值



卷積層其他參數

- Stride就是步幅，在卷積層工作的時候，Stride可以理解為每次滑動的單位。
 - Stride=1，每次只滑動1單位，能保證最好的細緻程度。
 - Stride=3，每次移動3單位，掃描訊息次數降低，但是因為處理次數變少，卷積層在掃描的時候工作會變快。

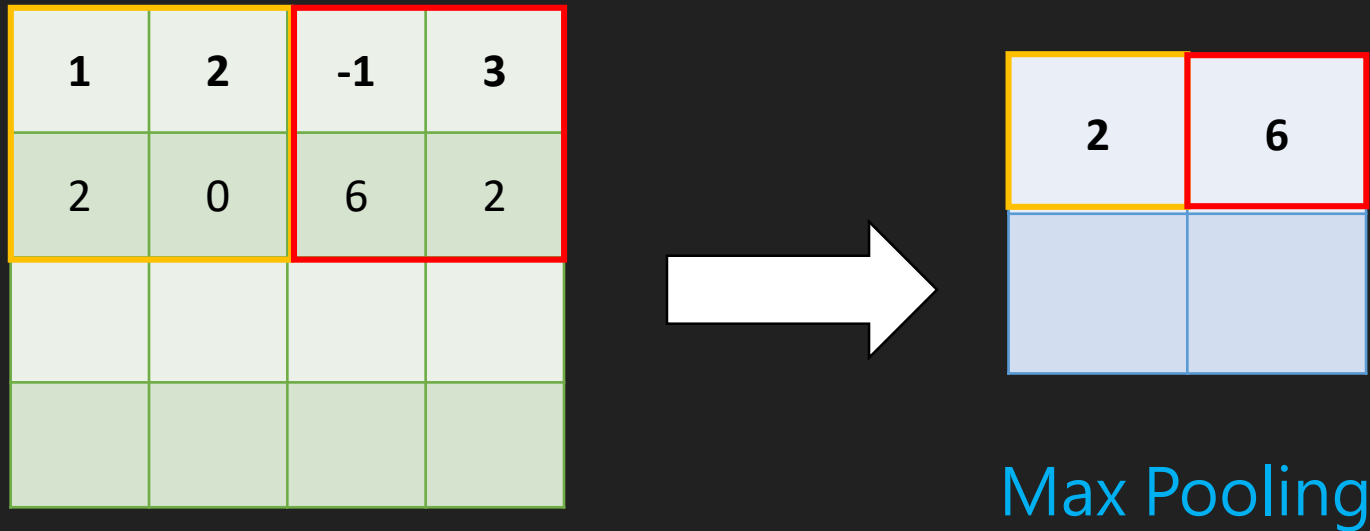


4. 池化層 Pooling Layer

Pooling Layer

- 池化層處理有兩種方式：Max Pooling、Mean Pooling
- Max Pooling：在前面輸出過來的資料上取最大值

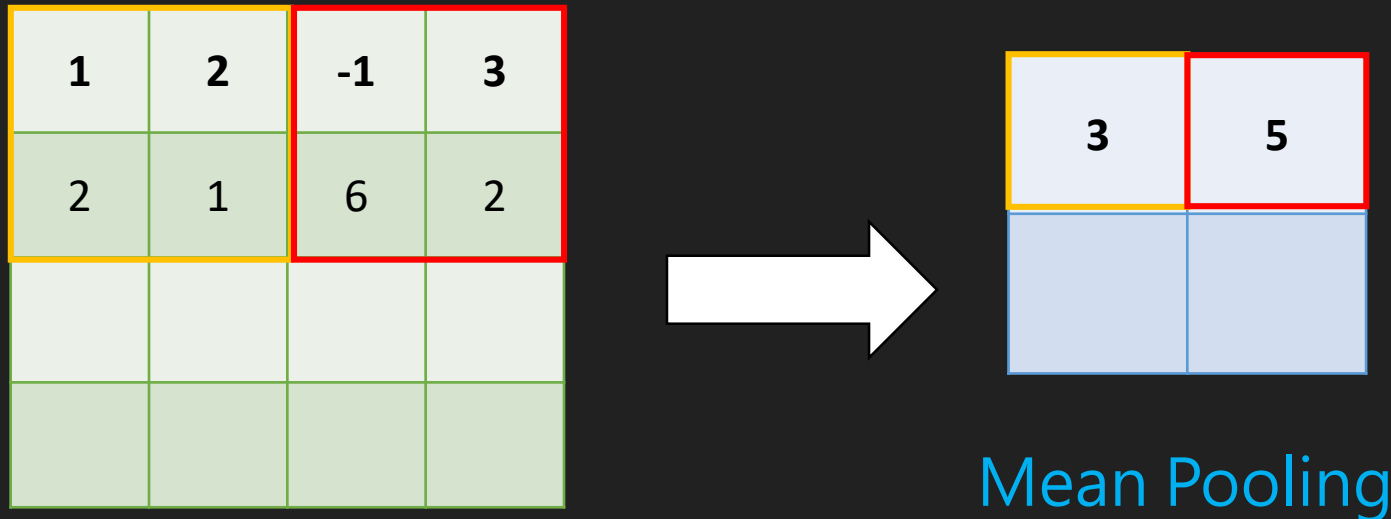
假設有一個Stride=2的2x2 Max Pooling Filter，對輸出過來的資料取最大值則我們可以得到2和6，再將其儲存至Max Pooling層中



Pooling Layer

- Mean Pooling : 在前面輸出過來的資料上取平均值。

假設有一個Stride=2的2x2 Mean Pooling Filter，對輸出過來的資料取平均值則我們可以得到2和0，再將其儲存至Mean Pooling層中

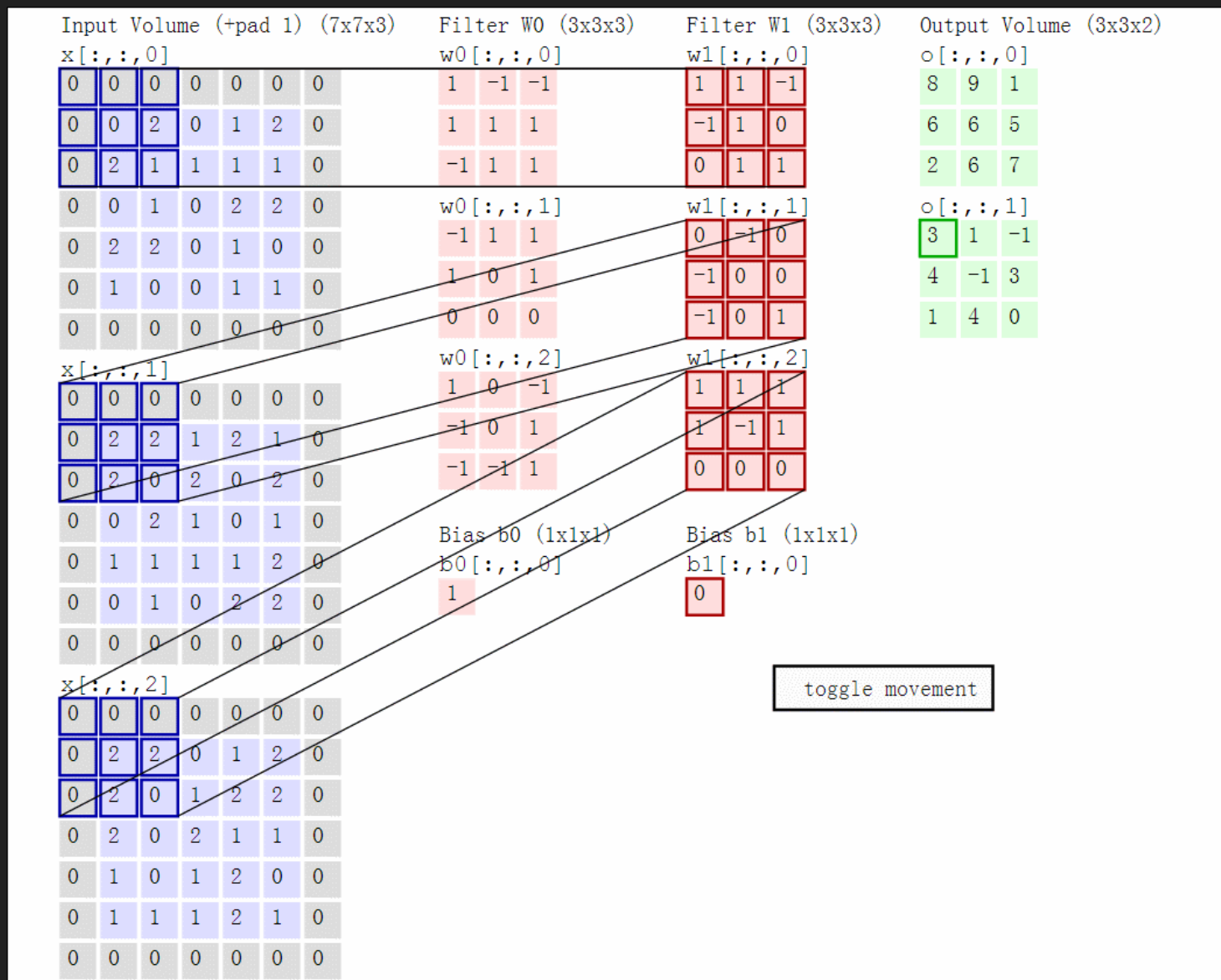


Pooling Layer

■ 池化層具有下列功能

- 第一：它又進行了一次特徵萃取，可以減少下一層資料的處理量
- 第二：由於這個特徵的萃取，能夠有更大的可能性進一步獲取更為抽象的訊息，從而防止過適(Overfitting)，或著說提高一定的廣義性

Convolution 運算



5. 典型CNN網路

典型CNN網路

- 目前世界上比較新的網路，由於運算能力提升而變得越來越複雜，使得整個網路不再呈現典型純粹的全連接神經網路、卷神經網路、或其他網路的獨有特點。
- VGG-16：VGG 就是 Visual Geometry Group 的縮寫，16指的是13個卷積層和3個全連接層的網路層。

典型CNN網路

- 每年，史丹佛都會舉辦一個比賽，叫作 ImageNet 視覺識別挑戰。

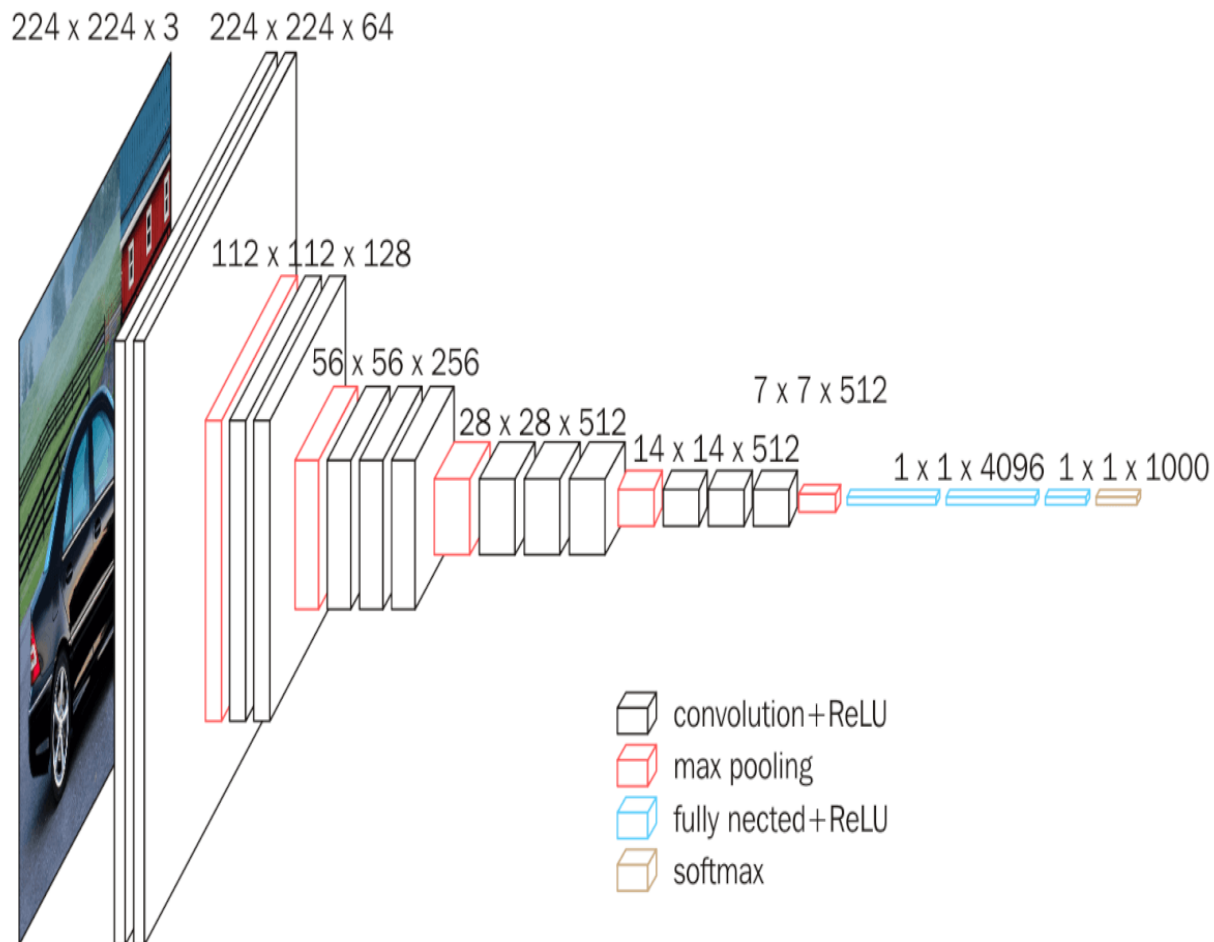
The Google logo, featuring the word "Google" in its characteristic multi-colored font (blue, red, yellow, blue, green, red).The Microsoft logo, consisting of a four-colored square icon (orange, green, blue, yellow) followed by the word "Microsoft" in a grey sans-serif font.The Baidu logo, featuring the word "Bai" in red, a blue paw print icon containing the word "du" in white, and the Chinese characters "百度" in red.

典型CNN網路

- Keras把冠軍都收錄進框架內，稱為Keras Applications

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters
Xception	88 MB	0.790	0.945	22,910,480
VGG16	528 MB	0.715	0.901	138,357,544
VGG19	549 MB	0.727	0.910	143,667,240
ResNet50	99 MB	0.759	0.929	25,636,712
InceptionV3	92 MB	0.788	0.944	23,851,784
InceptionResNetV2	215 MB	0.804	0.953	55,873,736
MobileNet	17 MB	0.665	0.871	4,253,864

典型CNN網路-VGG16

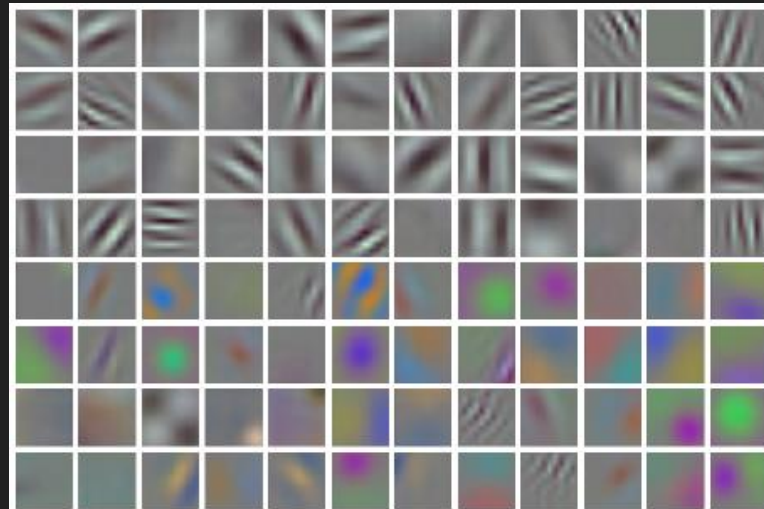


```
model = Sequential([
    Conv2D(64, (3, 3), input_shape=input_shape, padding='same',
          activation='relu'),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    Conv2D(256, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Conv2D(512, (3, 3), activation='relu', padding='same'),
    Conv2D(512, (3, 3), activation='relu', padding='same'),
    Conv2D(512, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Conv2D(512, (3, 3), activation='relu', padding='same'),
    Conv2D(512, (3, 3), activation='relu', padding='same'),
    Conv2D(512, (3, 3), activation='relu', padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Flatten(),
    Dense(4096, activation='relu'),
    Dense(4096, activation='relu'),
    Dense(1000, activation='softmax')
```

6. 圖片識別

圖片識別

- 將Feature Map 的內容進行視覺化，會看到類似這樣的一些圖案，觀察這些光斑我們可以得到下面幾個觀點
 - 1. 少量的噪聲對深度卷神經網路的分類影響是非常有限的
 - 2. 由於卷積神經網路取最大特徵值的方式使其廣義性更好，因為即使分類物件跟訓練樣本的特徵有一定差異，這種「模糊化」處理的結果會使得它們在較深的網路中有類似的結果。



7. 輸出層激勵函數-SOFTMAX

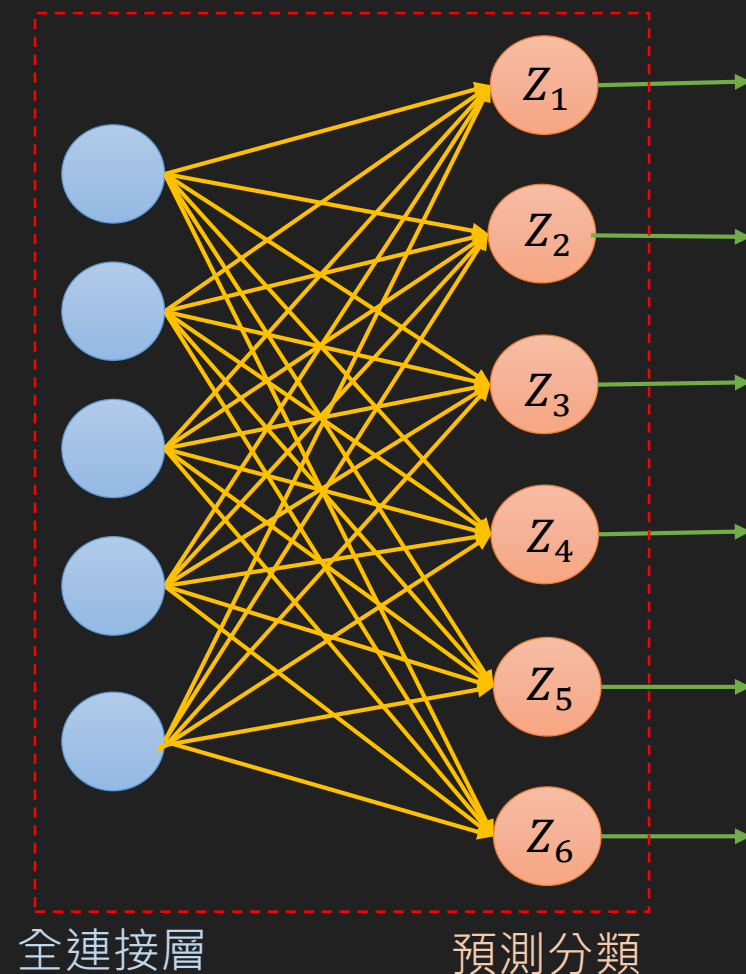
輸出層激勵函數-SOFTMAX

- 活化函數的輸出層(右圖)，每個預測分類的節點代表一個分類，這6個節點最多能表示6個分類的模型
- 任一節點代表的活化函數:

$$\sigma_i(z) = \frac{\exp(Z_i)}{\sum_{i=1}^n \exp(Z_i)}$$

i 是節點的下標次序，而 $Z_i = xw_i + b_i$

b_i 是由使用者制定



輸出層激勵函數-SOFTMAX

- 預測分類有這樣一個特性： $\sum_{i=1}^J \sigma_i(z) = 1$
也就是最後一層的每個節點輸出值總值為1
- 假設有一個訓練樣本和給分類標籤一個下標序號，對應的節點給1，其他給0。如果有6張不同的圖片，分別代表飛機、汽車、輪船、貓、狗、太陽按順序這些圖片分別應該被標記為

飛機	$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	汽車	$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	輪船	$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$	貓	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$	狗	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$	太陽	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$	預測分類輸出形式	$\begin{bmatrix} 0.001 \\ 0.030 \\ 0.620 \\ 0.101 \\ 0.020 \\ 0.219 \end{bmatrix}$
----	--	----	--	----	--	---	--	---	--	----	--	----------	--

8. 獨熱編碼 One-hot encoding

One-hot encoding

■ One-hot encoding 用一個向量的每一個維度來標識一種性質有無的方式

- 例如「性別」作為向量輸入的時候： $[1]$ 和 $[0]$

用One-hot encoding表示： $[1, 0]$ 和 $[0, 1]$

- 例如幾種不同的汽車類別用One-hot encoding表示：

轎車： $[1, 0, 0, 0, 0, 0]$

貨車： $[0, 0, 0, 1, 0, 0]$

皮卡： $[0, 1, 0, 0, 0, 0]$

大巴： $[0, 0, 0, 0, 1, 0]$

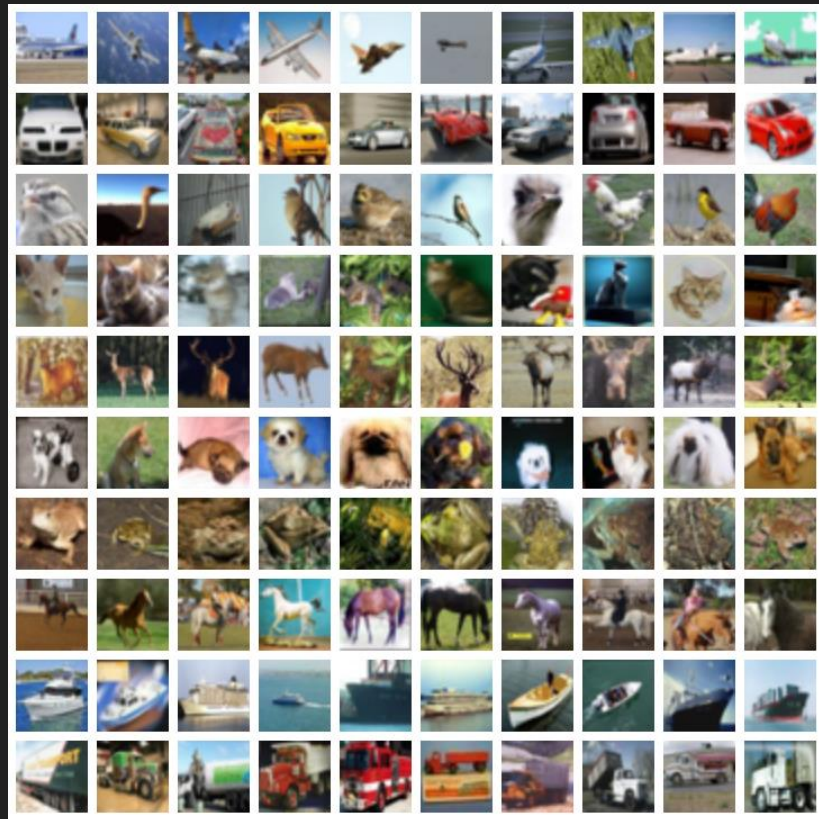
SUV： $[0, 0, 1, 0, 0, 0]$

其它： $[0, 0, 0, 0, 0, 1]$

9. CIFAR-10介紹

CIFAR-10資料集

- 全名: Cooperative Institute for Arctic Research
- 用於物體識別的資料集，也稱為CIFAR-10 Dataset



CIFAR-10資料集

- CIFAR-10是用於物體識別的資料集，是公開的資料集
整個資料集包含了60000張32 X 32像素的彩色照片
其中50000張是訓練集，10000為測試集
- 10個不同類別的標籤：
分別為airplane、automobile、bird、cat、deer、dog、frog、
horse、ship、truck

CIFAR-10官方網站

■ <https://www.cs.toronto.edu/~kriz/cifar.html>

[< Back to Alex Krizhevsky's home page](#)

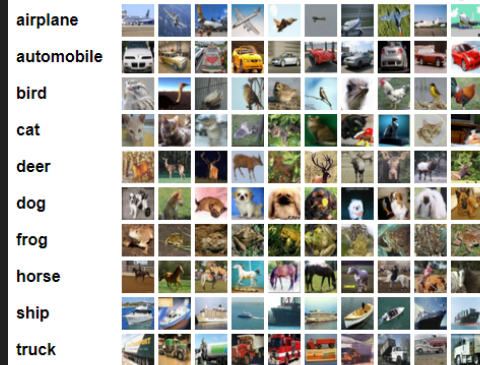
The CIFAR-10 and CIFAR-100 are labeled subsets of the [80 million tiny images](#) dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version	Size	md5sum
CIFAR-10 python version	163 MB	c58f30108f718f92721af3b95e74349a
CIFAR-10 Matlab version	175 MB	70270af85842c9e89bb428ec9976c926
CIFAR-10 binary version (suitable for C programs)	162 MB	c32a1d4ab5d03f1284b67883e8d87530

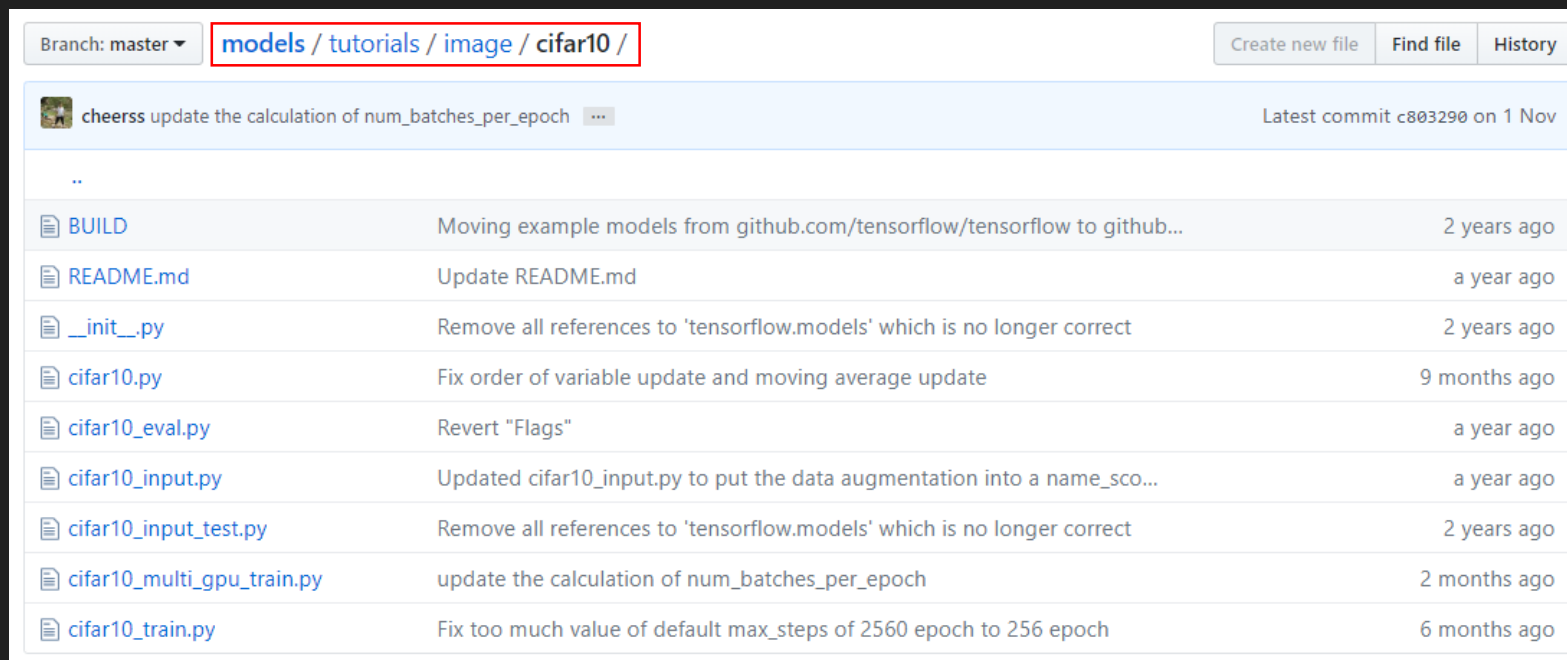
Baseline results

You can find some baseline replicable results on this dataset [on the project page for cuda-convnet](#). These results were obtained with a convolutional neural network. Briefly, they are 18% test error without data augmentation and 11% with. Additionally, [Jasper Snoek](#) has a [new paper](#) in which he used Bayesian hyperparameter optimization to find nice settings of the weight decay and other hyperparameters, which allowed him to obtain a test error rate of 15% (without data augmentation) using the architecture of the net that got 18%.

7. 牛刀小試-卷積神經網路做圖片分類

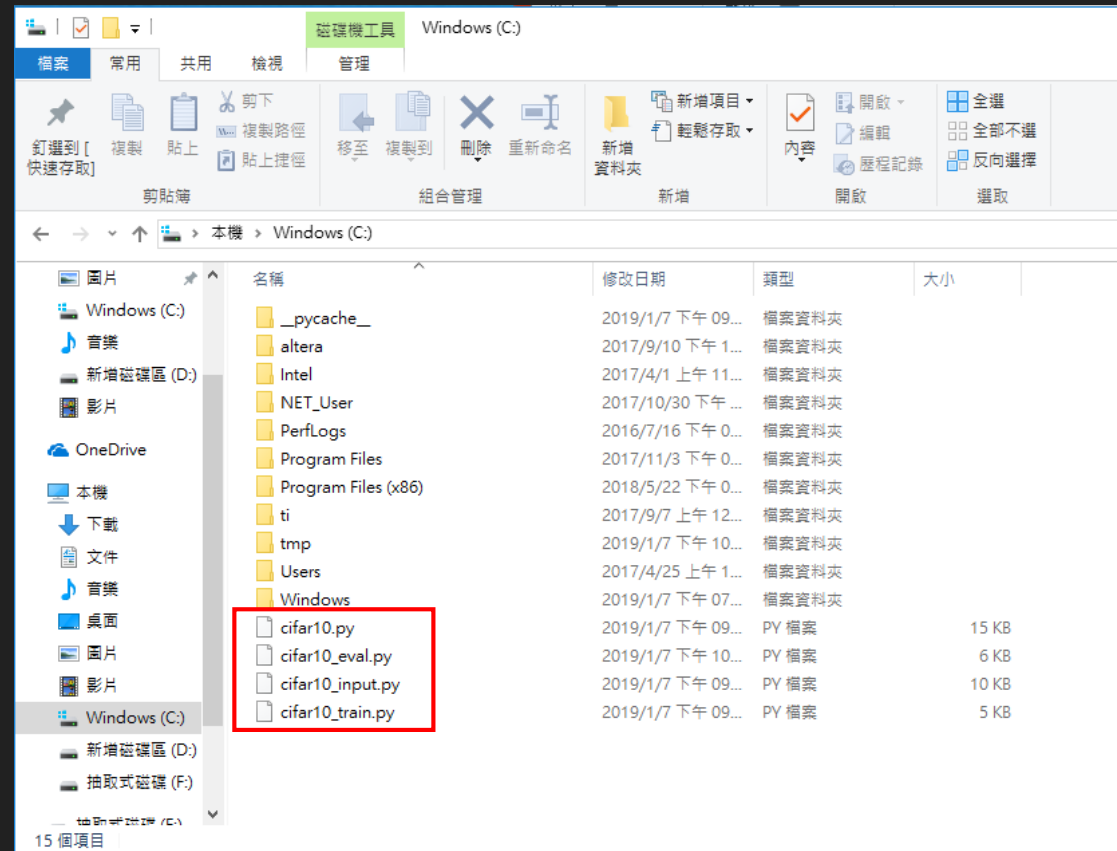
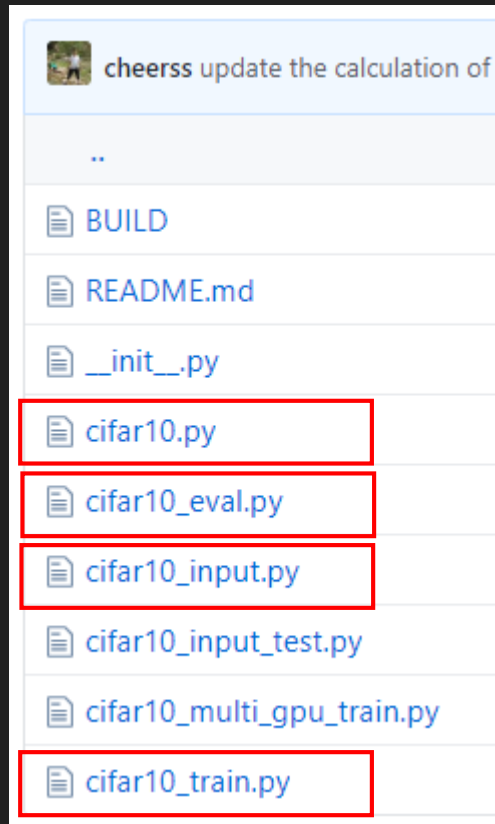
牛刀小試-卷積神經網路做圖片分類

- 卷積神經網路(Convolution Neural Network, CNN)完成 CIFAR-10 資料集物體辨識的工作
- TensorFlow 官方的 GitHub 上面有提供此次實驗的程式碼
- <https://github.com/tensorflow/models/tree/master/tutorials/image/cifar10>



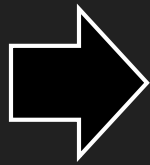
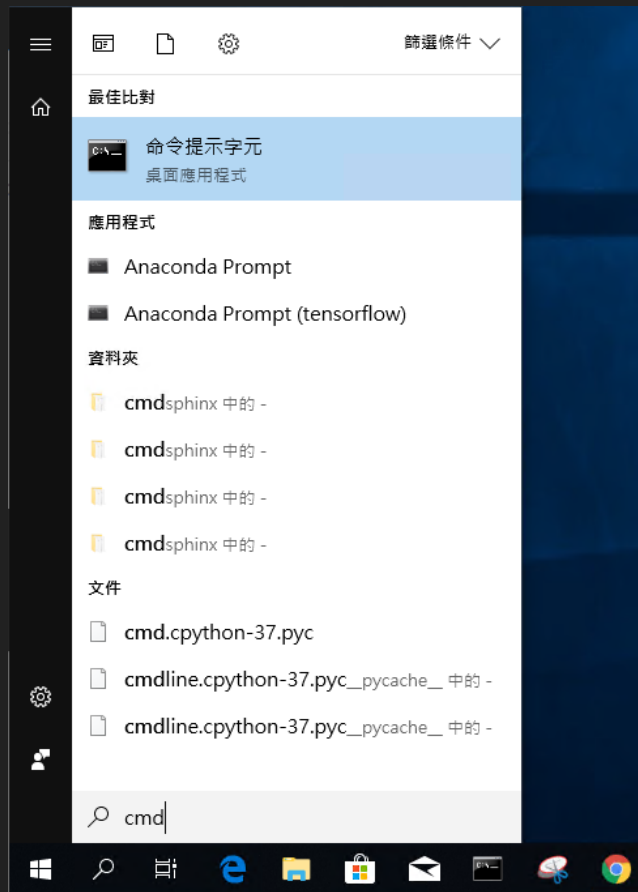
牛刀小試-卷積神經網路做圖片分類

- 下載檔案 cifar10.py 、 cifar10_input.py 、 cifar10_train.py 、 cifar10_eval.py 至 C:\ 目錄之下



牛刀小試-卷積神經網路做圖片分類

- 在搜尋輸入 cmd，開啟命令提示字元

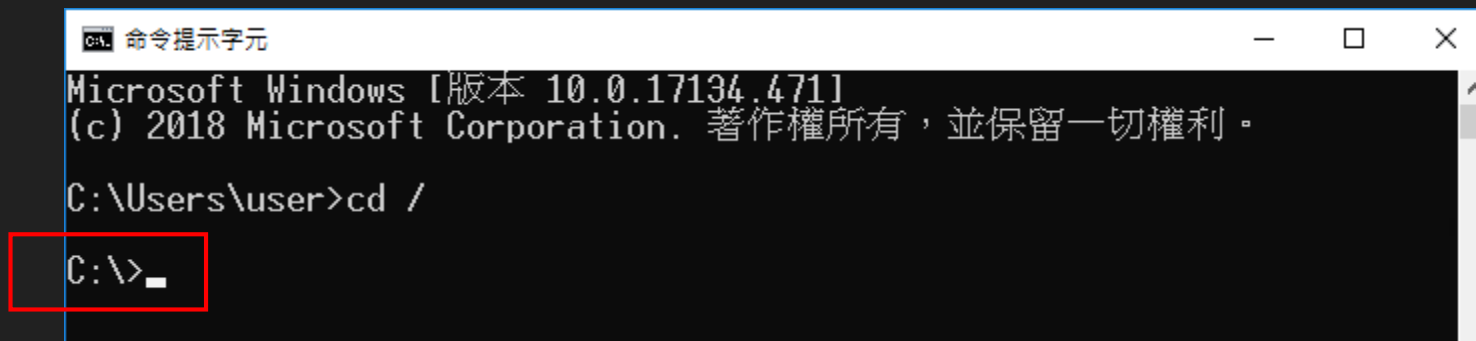


牛刀小試-卷積神經網路做圖片分類

- 在 cmd 視窗，輸入以下指令

```
> cd /
```

- 工作路徑由 C:\Users\user 變成 C:\



A screenshot of a Windows Command Prompt window. The title bar reads "命令提示字元". The window content shows the following text: "Microsoft Windows [版本 10.0.17134.471]", "(c) 2018 Microsoft Corporation. 著作權所有，並保留一切權利。", "C:\Users\user>cd /", and "C:\>". The "C:\>" prompt is highlighted with a red rectangular box.

牛刀小試-卷積神經網路做圖片分類

- 在 cmd 視窗，輸入以下指令

> `activate tensorflow` # 之前已經建立該虛擬環境



```
命令提示字元
C:\>activate tensorflow
(tensorflow) C:\>
```

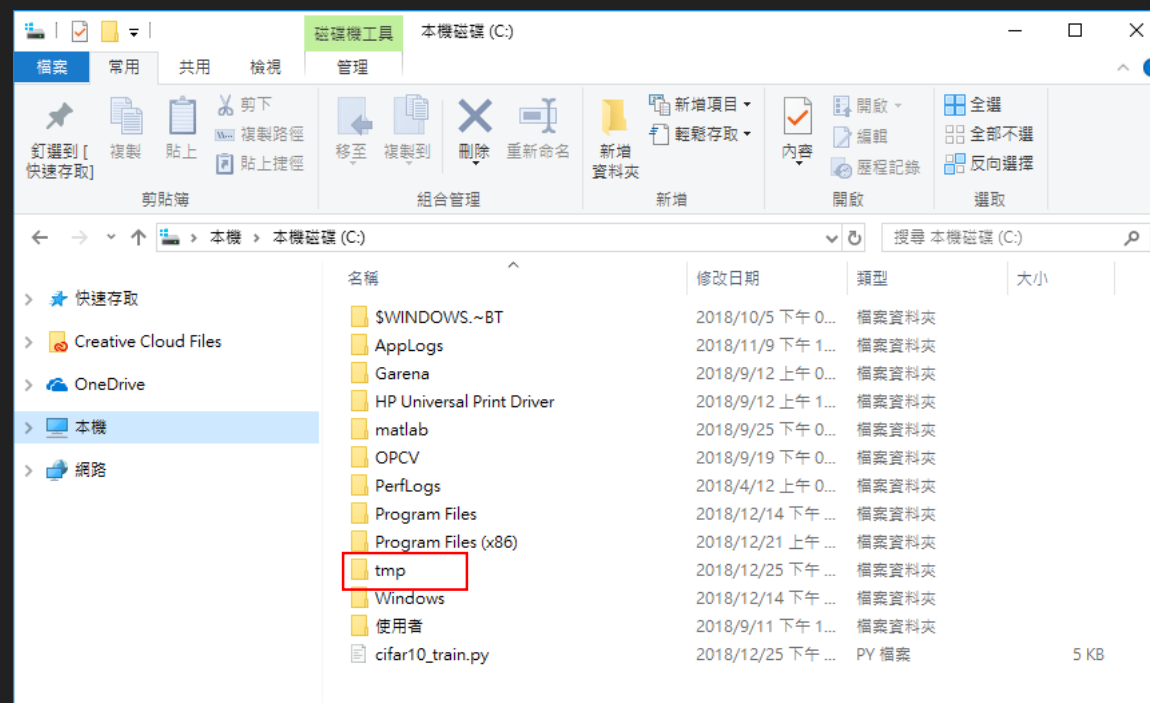
牛刀小試-卷積神經網路做圖片分類

- 在 cmd 視窗，輸入以下指令

> mkdir tmp # 建立目錄

```
命令提示字元
(tensorflow) C:\>mkdir tmp
(tensorflow) C:\>_
```

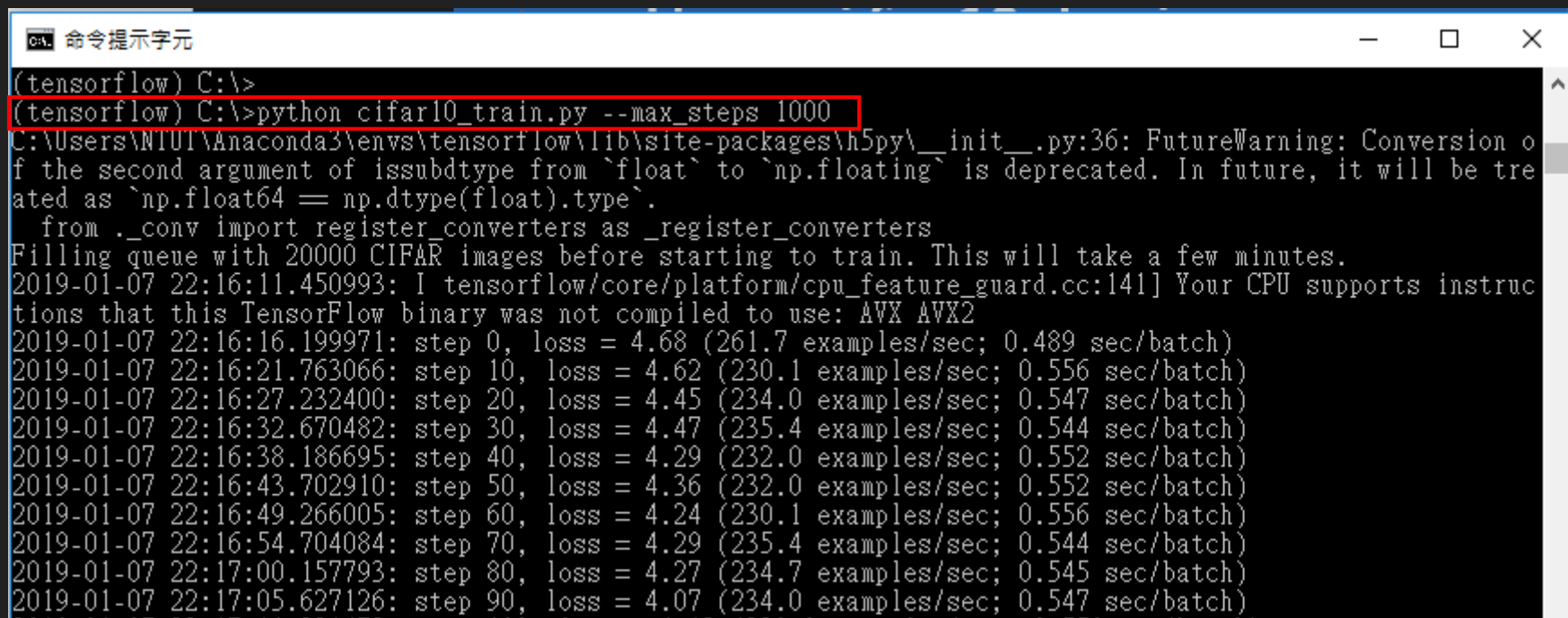
tmp 為放置輸入資料的目錄



牛刀小試-卷積神經網路做圖片分類

- 在 cmd 視窗，輸入以下指令

> `python cifar10_train.py --max_steps 1000`



```
命令提示字元
(tensorflow) C:\>
(tensorflow) C:\>python cifar10_train.py --max_steps 1000
C:\Users\N1UT\Anaconda3\envs\tensorflow\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of
f the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be tre
ated as `np.float64 = np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
Filling queue with 20000 CIFAR images before starting to train. This will take a few minutes.
2019-01-07 22:16:11.450993: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instruc
tions that this TensorFlow binary was not compiled to use: AVX AVX2
2019-01-07 22:16:16.199971: step 0, loss = 4.68 (261.7 examples/sec; 0.489 sec/batch)
2019-01-07 22:16:21.763066: step 10, loss = 4.62 (230.1 examples/sec; 0.556 sec/batch)
2019-01-07 22:16:27.232400: step 20, loss = 4.45 (234.0 examples/sec; 0.547 sec/batch)
2019-01-07 22:16:32.670482: step 30, loss = 4.47 (235.4 examples/sec; 0.544 sec/batch)
2019-01-07 22:16:38.186695: step 40, loss = 4.29 (232.0 examples/sec; 0.552 sec/batch)
2019-01-07 22:16:43.702910: step 50, loss = 4.36 (232.0 examples/sec; 0.552 sec/batch)
2019-01-07 22:16:49.266005: step 60, loss = 4.24 (230.1 examples/sec; 0.556 sec/batch)
2019-01-07 22:16:54.704084: step 70, loss = 4.29 (235.4 examples/sec; 0.544 sec/batch)
2019-01-07 22:17:00.157793: step 80, loss = 4.27 (234.7 examples/sec; 0.545 sec/batch)
2019-01-07 22:17:05.627126: step 90, loss = 4.07 (234.0 examples/sec; 0.547 sec/batch)
```

牛刀小試-卷積神經網路做圖片分類

- 出現以下圖式代表訓練成功

```
cmd 命令提示字元
2019-01-07 22:22:48.361430: step 710, loss = 2.68 (237.4 examples/sec; 0.539 sec/batch)
2019-01-07 22:22:53.768258: step 720, loss = 2.84 (236.7 examples/sec; 0.541 sec/batch)
2019-01-07 22:22:59.206341: step 730, loss = 3.16 (235.4 examples/sec; 0.544 sec/batch)
2019-01-07 22:23:04.660047: step 740, loss = 2.76 (234.7 examples/sec; 0.545 sec/batch)
2019-01-07 22:23:10.191886: step 750, loss = 2.69 (231.4 examples/sec; 0.553 sec/batch)
2019-01-07 22:23:15.614342: step 760, loss = 2.70 (236.1 examples/sec; 0.542 sec/batch)
2019-01-07 22:23:21.005546: step 770, loss = 2.68 (237.4 examples/sec; 0.539 sec/batch)
2019-01-07 22:23:26.443639: step 780, loss = 2.56 (235.4 examples/sec; 0.544 sec/batch)
2019-01-07 22:23:31.866079: step 790, loss = 2.64 (236.1 examples/sec; 0.542 sec/batch)
2019-01-07 22:23:37.366665: step 800, loss = 2.76 (232.7 examples/sec; 0.550 sec/batch)
2019-01-07 22:23:42.773493: step 810, loss = 2.85 (236.7 examples/sec; 0.541 sec/batch)
2019-01-07 22:23:48.227202: step 820, loss = 2.68 (234.7 examples/sec; 0.545 sec/batch)
2019-01-07 22:23:53.634028: step 830, loss = 2.66 (236.7 examples/sec; 0.541 sec/batch)
2019-01-07 22:23:59.072109: step 840, loss = 2.79 (235.4 examples/sec; 0.544 sec/batch)
2019-01-07 22:24:04.525817: step 850, loss = 2.69 (234.7 examples/sec; 0.545 sec/batch)
2019-01-07 22:24:09.932657: step 860, loss = 2.55 (236.7 examples/sec; 0.541 sec/batch)
2019-01-07 22:24:15.323844: step 870, loss = 2.60 (237.4 examples/sec; 0.539 sec/batch)
2019-01-07 22:24:20.746299: step 880, loss = 2.46 (236.1 examples/sec; 0.542 sec/batch)
2019-01-07 22:24:26.121873: step 890, loss = 2.53 (238.1 examples/sec; 0.538 sec/batch)
2019-01-07 22:24:31.606834: step 900, loss = 2.62 (233.4 examples/sec; 0.548 sec/batch)
2019-01-07 22:24:37.044915: step 910, loss = 2.68 (235.4 examples/sec; 0.544 sec/batch)
2019-01-07 22:24:42.451742: step 920, loss = 2.73 (236.7 examples/sec; 0.541 sec/batch)
2019-01-07 22:24:48.014837: step 930, loss = 2.45 (230.1 examples/sec; 0.556 sec/batch)
2019-01-07 22:24:53.562304: step 940, loss = 2.60 (230.7 examples/sec; 0.555 sec/batch)
2019-01-07 22:24:58.969131: step 950, loss = 2.37 (236.7 examples/sec; 0.541 sec/batch)
2019-01-07 22:25:04.407213: step 960, loss = 2.38 (235.4 examples/sec; 0.544 sec/batch)
2019-01-07 22:25:09.876547: step 970, loss = 2.63 (234.0 examples/sec; 0.547 sec/batch)
2019-01-07 22:25:15.392762: step 980, loss = 2.60 (232.0 examples/sec; 0.552 sec/batch)
2019-01-07 22:25:20.799589: step 990, loss = 2.29 (236.7 examples/sec; 0.541 sec/batch)

(tensorflow) C:\>
```

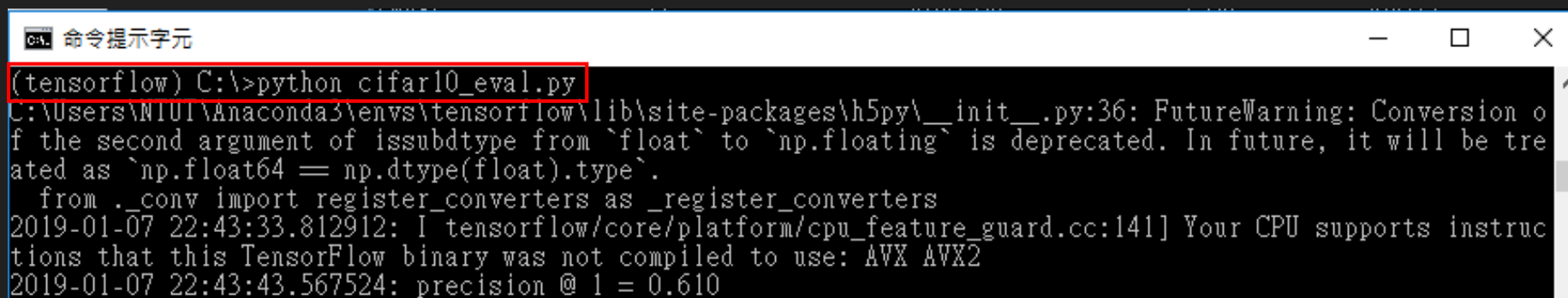

牛刀小試-卷積神經網路做圖片分類

- 接下來要對剛剛訓練的模型，進行準確率評估
- 在 cmd 視窗，輸入以下指令

> python cifar10_eval.py

- 準確率 precision = 0.61

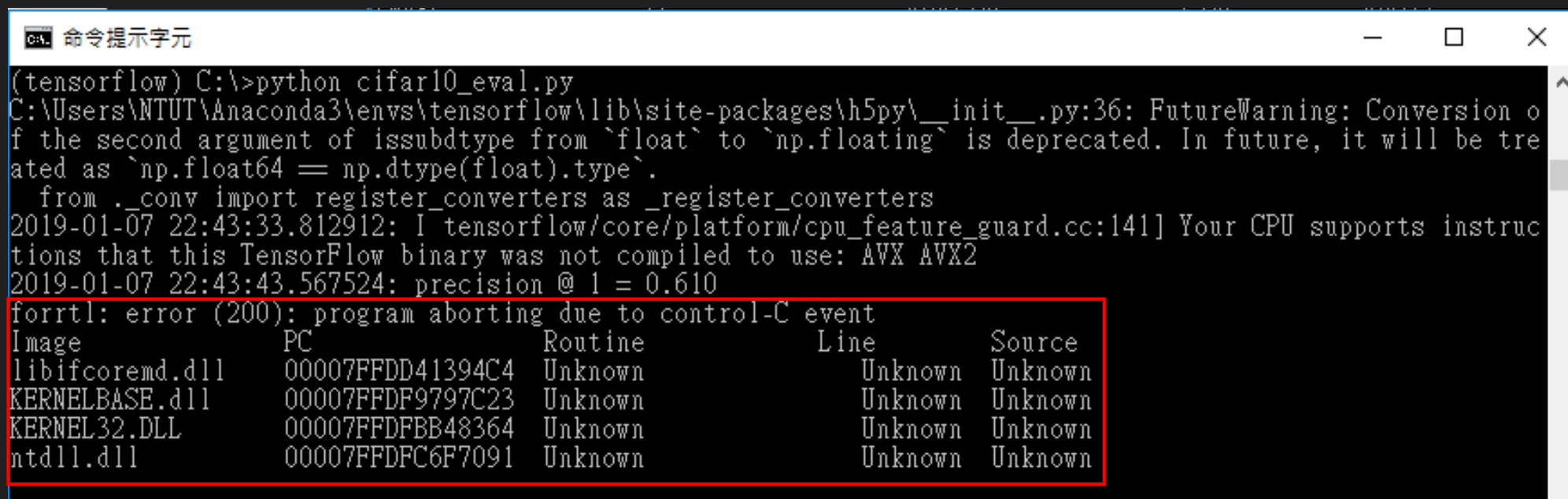
Issue: (max_steps ↑ , 準確率 ↑)???



```
命令提示字元
(tensorflow) C:\>python cifar10_eval.py
C:\Users\NIUI\Anaconda3\envs\tensorflow\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of
the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be tre
ated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
2019-01-07 22:43:33.812912: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instruc
tions that this TensorFlow binary was not compiled to use: AVX AVX2
2019-01-07 22:43:43.567524: precision @ 1 = 0.610
```

牛刀小試-卷積神經網路做圖片分類

- 按 Ctrl + C ， 離開評估腳本



```
(tensorflow) C:\>python cifar10_eval.py
C:\Users\NTUT\Anaconda3\envs\tensorflow\lib\site-packages\h5py\__init__.py:36: FutureWarning: Conversion of
f the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be tre
ated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
2019-01-07 22:43:33.812912: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instruc
tions that this TensorFlow binary was not compiled to use: AVX AVX2
2019-01-07 22:43:43.567524: precision @ 1 = 0.610
forrtl: error (200): program aborting due to control-C event
Image                PC                Routine          Line           Source
libifcoremd.dll      00007FFDD41394C4  Unknown         Unknown        Unknown
KERNELBASE.dll       00007FFDF9797C23  Unknown         Unknown        Unknown
KERNEL32.DLL         00007FFDFB48364  Unknown         Unknown        Unknown
ntdll.dll             00007FFDFC6F7091  Unknown         Unknown        Unknown
```

9. cifar10_train.py 程式碼解析

cifar10_train.py 程式碼解析

```
def train():
```

```
    """Train CIFAR-10 for a number of steps."""
```

```
    with tf.Graph().as_default():
```

使用預設圖

```
        global_step = tf.train.get_or_create_global_step()
```

```
        # Get images and labels for CIFAR-10.
```

```
        # Force input pipeline to CPU:0 to avoid operations sometimes ending up on
```

```
        # GPU and resulting in a slow down.
```

```
        with tf.device('/cpu:0'):
```

```
            images, labels = cifar10.distorted_inputs()
```

獲取訓練資料和其對應的標籤

```
        # Build a Graph that computes the logits predictions from the
```

```
        # inference model.
```

```
        logits = cifar10.inference(images)
```

```
        # Calculate loss.
```

```
        loss = cifar10.loss(logits, labels)
```

```
        # Build a Graph that trains the model with one batch of examples and
```

```
        # updates the model parameters.
```

```
        train_op = cifar10.train(loss, global_step)
```

建立網路Op、loss OP、訓練Op

cifar10_train.py 程式碼解析

```
def begin(self):
```

```
    self._step = -1
```

```
    self._start_time = time.time()
```

開始訓練之前設定當前步數

```
def before_run(self, run_context):
```

```
    self._step += 1
```

每次執行之前更新當前步數

```
    return tf.train.SessionRunArgs(loss) # Asks for loss value.
```

執行一個step，傳進loss OP

```
def after_run(self, run_context, run_values):
```

```
    if self._step % FLAGS.log_frequency == 0:
```

```
        current_time = time.time()
```

```
        duration = current_time - self._start_time
```

```
        self._start_time = current_time
```

```
        loss_value = run_values.results
```

loss Op的返回結果

```
        examples_per_sec = FLAGS.log_frequency * FLAGS.batch_size / duration
```

```
        sec_per_batch = float(duration / FLAGS.log_frequency)
```

```
        format_str = ('%s: step %d, loss = %.2f (%.1f examples/sec; %.3f '
                       'sec/batch)')
```

```
        print (format_str % (datetime.now(), self._step, loss_value,
                              examples_per_sec, sec_per_batch))
```

```
step 55010, loss = 0.80
step 55020, loss = 0.80
step 55030, loss = 0.86
step 55040, loss = 0.76
step 55050, loss = 0.80
step 55060, loss = 0.71
step 55070, loss = 0.69
step 55080, loss = 0.68
step 55090, loss = 0.58
step 55100, loss = 0.62
step 55110, loss = 0.70
step 55120, loss = 0.82
```

執行時每10個step輸出訊息

cifar10_train.py 程式碼解析

使用預設圖

```
with tf.train.MonitoredTrainingSession(  
    checkpoint_dir=FLAGS.train_dir,  
    hooks=[tf.train.StopAtStepHook(last_step=FLAGS.max_steps),  
          tf.train.NanTensorHook(loss),  
          _LoggerHook()],  
    config=tf.ConfigProto(  
        log_device_placement=FLAGS.log_device_placement)) as mon_sess:  
    while not mon_sess.should_stop():  
        mon_sess.run(train_op)
```

這裡使用MonitoredTrainingSession可以設定Hook函數在開始訓練之前，以及執行過程中設定回調函數處理變數，輸出訊息

開始訓練，傳入train Op直到
FLAGS.max_steps停止上方程式碼有設定
tf.train.StopAtStepHook(last_step=FLAGS.
max_steps)

8. cifar10.py 程式碼解析

cifar10.py 程式碼解析

```
# Global constants describing the CIFAR-10 data set.
IMAGE_SIZE = cifar10_input.IMAGE_SIZE
NUM_CLASSES = cifar10_input.NUM_CLASSES
NUM_EXAMPLES_PER_EPOCH_FOR_TRAIN = cifar10_input.NUM_EXAMPLES_PER_EPOCH_FOR_TRAIN
NUM_EXAMPLES_PER_EPOCH_FOR_EVAL = cifar10_input.NUM_EXAMPLES_PER_EPOCH_FOR_EVAL
```

CIFAR10 dataset 圖片大小，分類
數目，訓練和驗證資料每個Epoch
的資料數量

```
# Constants describing the training process.
MOVING_AVERAGE_DECAY = 0.9999 # The decay to use for the moving average.
NUM_EPOCHS_PER_DECAY = 350.0 # Epochs after which learning rate decays.
LEARNING_RATE_DECAY_FACTOR = 0.1 # Learning rate decay factor.
INITIAL_LEARNING_RATE = 0.1 # Initial learning rate.
```

訓練參數

```
# If a model is trained with multiple GPUs, prefix all Op names with tower_name
# to differentiate the operations. Note that this prefix is removed from the
# names of the summaries when visualizing a model.
TOWER_NAME = 'tower'
```

```
DATA_URL = 'https://www.cs.toronto.edu/~kriz/cifar-10-binary.tar.gz'
```

CIFAR10 dataset 下載地址

cifar10.py 程式碼解析

```
def _variable_on_cpu(name, shape, initializer):  
    """Helper to create a Variable stored on CPU memory.
```

Args:

name: name of the variable
shape: list of ints
initializer: initializer for Variable

Returns:

Variable Tensor

"""

```
with tf.device('/cpu:0'):
```

```
    dtype = tf.float16 if FLAGS.use_fp16 else tf.float32
```

```
    var = tf.get_variable(name, shape, initializer=initializer, dtype=dtype)
```

```
    return var
```

指定在cpu: 0上建立變數

獲取一個已經存在的變數或者
在建立一個新的

cifar10.py 程式碼解析

```
def _variable_with_weight_decay(name, shape, stddev, wd):  
    """Helper to create an initialized Variable with weight decay.  
  
    Note that the Variable is initialized with a truncated normal distribution.  
    A weight decay is added only if one is specified.  
  
    Args:  
        name: name of the variable  
        shape: list of ints  
        stddev: standard deviation of a truncated Gaussian  
        wd: add L2Loss weight decay multiplied by this float. If None, weight  
            decay is not added for this Variable.  
  
    Returns:  
        Variable Tensor  
    """  
    dtype = tf.float16 if FLAGS.use_fp16 else tf.float32  
    var = _variable_on_cpu( -----> 指定在cpu上建立變數  
        name,  
        shape,  
        tf.truncated_normal_initializer(stddev=stddev, dtype=dtype))  
    if wd is not None:  
        weight_decay = tf.multiply(tf.nn.l2_loss(var), wd, name='weight_loss')  
        tf.add_to_collection('losses', weight_decay)  
    return var
```

建立weight變數並儲存權值衰減(weight decay)，這是一種防止過適的手段，在每一次更新後，權值w會乘以一個衰減係數，就是這個weight decay，通常是一小略小於1的數字，以壓制w值的大小。關於位什麼壓制會有比較好的防止過適後面我們也會討論。

cifar10.py 程式碼解析

```
def inference(images):  
    """Build the CIFAR-10 model.  
  
    Args:  
        images: Images returned from distorted_inputs() or inputs().  
    # conv1  
    with tf.variable_scope('conv1') as scope:  
        kernel = _variable_with_weight_decay('weights',  
                                             shape=[5, 5, 3, 64],  
                                             stddev=5e-2,  
                                             wd=None)  
        conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding='SAME')  
        biases = _variable_on_cpu('biases', [64], tf.constant_initializer(0.0))  
        pre_activation = tf.nn.bias_add(conv, biases)  
        conv1 = tf.nn.relu(pre_activation, name=scope.name)  
        _activation_summary(conv1)  
  
    # pool1  
    pool1 = tf.nn.max_pool(conv1, ksize=[1, 3, 3, 1], strides=[1, 2, 2, 1],  
                           padding='SAME', name='pool1')  
    # norm1  
    norm1 = tf.nn.lrn(pool1, 4, bias=1.0, alpha=0.001 / 9.0, beta=0.75,  
                     name='norm1')
```

建立變數過濾器

建立2-D捲基層OP

建立變數偏向，在CPU上

添加偏移量

過活化函數

池化1

正規化1

cifar10.py 程式碼解析

```
with tf.variable_scope('local3') as scope:
```

```
    # Move everything into depth so we can perform a single matrix multiply.
```

```
    reshape = tf.reshape(pool2, [images.get_shape().as_list()[0], -1])
```

```
    dim = reshape.get_shape()[1].value
```

```
    weights = _variable_with_weight_decay('weights', shape=[dim, 384],  
                                           stddev=0.04, wd=0.004)
```

```
    biases = _variable_on_cpu('biases', [384], tf.constant_initializer(0.1))
```

```
    local3 = tf.nn.relu(tf.matmul(reshape, weights) + biases, name=scope.name)
```

```
    _activation_summary(local3)
```

把batch上的每一條資料平鋪
成1-D，因為要做全連階層

隨便獲取batch上的一條資料
的維度，其他全部一樣

cifar10.py 程式碼解析

```
def train(total_loss, global_step):  
    # Variables that affect learning rate.  
    num_batches_per_epoch = NUM_EXAMPLES_PER_EPOCH_FOR_TRAIN / FLAGS.batch_size  
    decay_steps = int(num_batches_per_epoch * NUM_EPOCHS_PER_DECAY)  
  
    # Decay the learning rate exponentially based on the number of steps.  
    lr = tf.train.exponential_decay(INITIAL_LEARNING_RATE,  
                                   global_step,  
                                   decay_steps,  
                                   LEARNING_RATE_DECAY_FACTOR,  
                                   staircase=True)  
    tf.summary.scalar('learning_rate', lr)  
  
    # Generate moving averages of all losses and associated summaries.  
    loss_averages_op = _add_loss_summaries(total_loss)  
  
    # Compute gradients.  
    with tf.control_dependencies([loss_averages_op]):  
        opt = tf.train.GradientDescentOptimizer(lr)  
        grads = opt.compute_gradients(total_loss)  
  
    # Apply gradients.  
    apply_gradient_op = opt.apply_gradients(grads, global_step=global_step)
```

學習率在學習過程中指數下降，慢慢變小

運算梯度依賴loss_averages_op就是先運算loss_averages_op在運算梯度

把運算完的梯度放回去

cifar10.py 程式碼解析

```
# Add histograms for trainable variables.
for var in tf.trainable_variables():
    tf.summary.histogram(var.op.name, var)
```

```
# Add histograms for gradients.
for grad, var in grads:
    if grad is not None:
        tf.summary.histogram(var.op.name + '/gradients', grad)
```

```
# Track the moving averages of all trainable variables.
```

```
variable_averages = tf.train.ExponentialMovingAverage(
    MOVING_AVERAGE_DECAY, global_step)
```

```
with tf.control_dependencies([apply_gradient_op]):
```

```
    variables_averages_op = variable_averages.apply(tf.trainable_variables())
```

```
return variables_averages_op
```

運算訓練過程中所有變數的移動平均值

運算train_op時需要先運算apply_gradient_op、
variables_averages_op

cifar10.py 程式碼解析

```
def maybe_download_and_extract():
    """Download and extract the tarball from Alex's website."""
    dest_directory = FLAGS.data_dir
    if not os.path.exists(dest_directory):
        os.makedirs(dest_directory)
    filename = DATA_URL.split('/')[-1]
    filepath = os.path.join(dest_directory, filename)
    if not os.path.exists(filepath):
        def _progress(count, block_size, total_size):
            sys.stdout.write('\r>> Downloading %s %.1f%%' % (filename,
                float(count * block_size) / float(total_size) * 100.0))
            sys.stdout.flush()
        filepath, _ = urllib.request.urlretrieve(DATA_URL, filepath, _progress)
        print()
        statinfo = os.stat(filepath)
        print('Successfully downloaded', filename, statinfo.st_size, 'bytes.')
    extracted_dir_path = os.path.join(dest_directory, 'cifar-10-batches-bin')
    if not os.path.exists(extracted_dir_path):
        tarfile.open(filepath, 'r:gz').extractall(dest_directory)
```

下載資料解壓縮資料

-END-