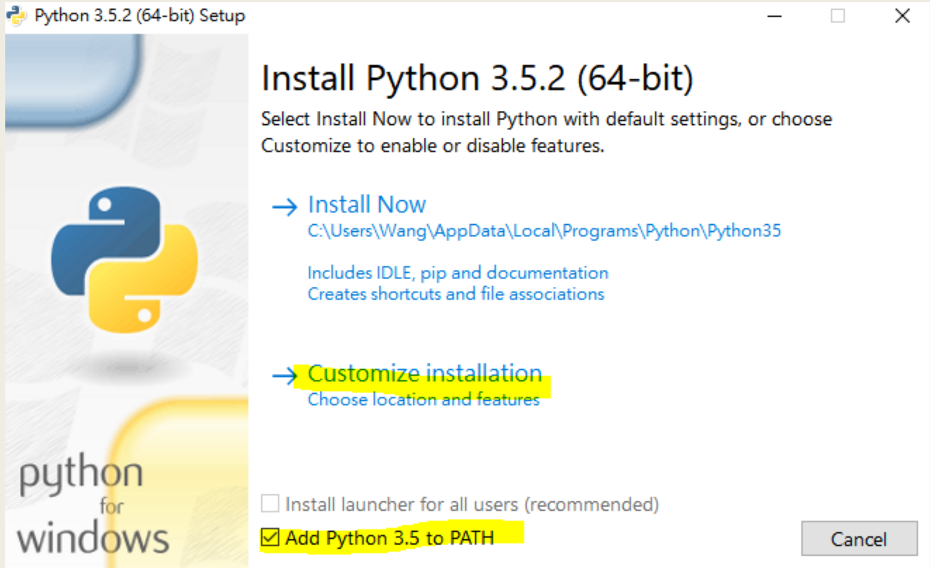
數位系統導論實驗
Lab2 TensorFlow & MNIST Dataset

課程目標

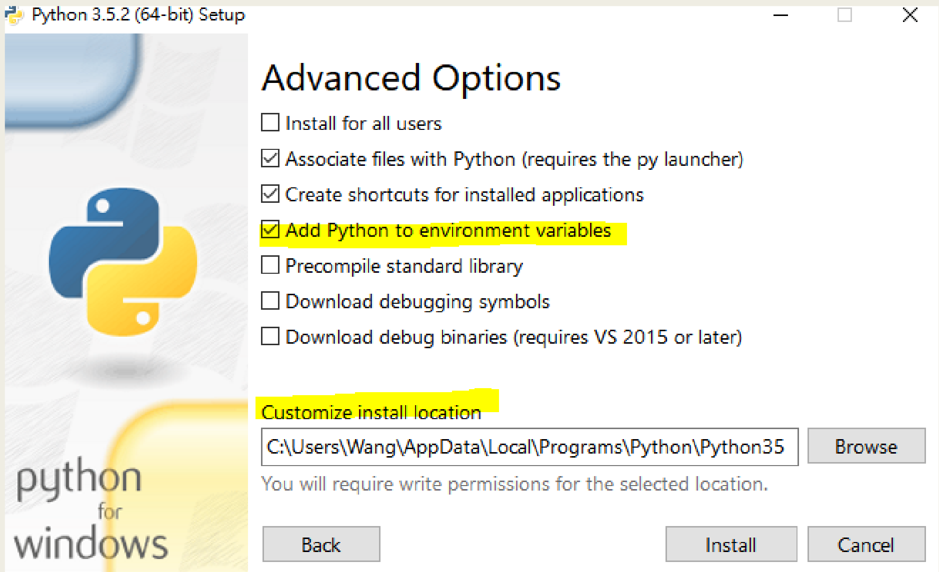
- 練習使用 TensorFlow 完成 DNN 之架構設計，並將 Lab1 設計之 UI 加入手寫數字辨識功能

實驗環境 – Python

- 此次 TensorFlow 在 Python 上運行，請同學至 Python 官網下載並安裝程式
- 因 TensorFlow 有限制 Python 版本，同學安裝時請避免3.5.x與3.6.x以外的版本



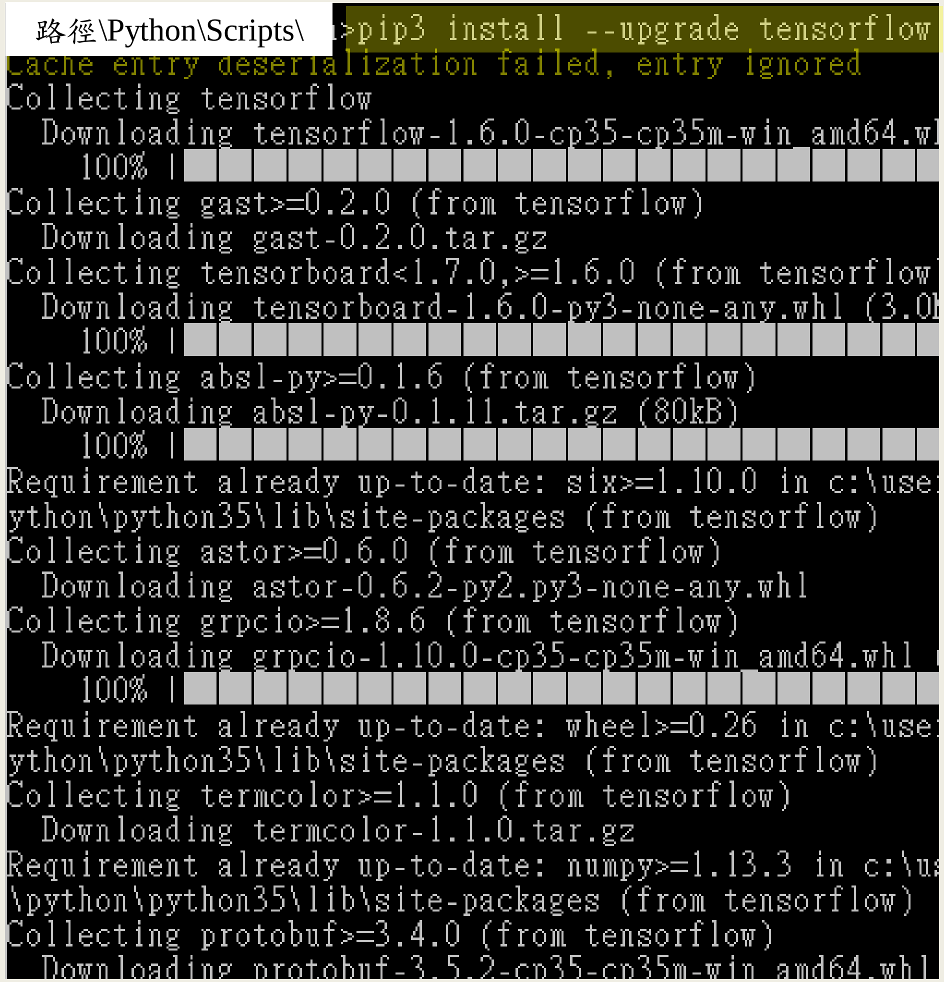
安裝設置一



安裝設置二

實驗環境 – TensorFlow (1/2)

- TensorFlow 是用於機器學習的開源軟體庫，同學可至官網了解詳細資訊
- 同學請於 Python 路徑下 Scripts 資料夾開啟 command，輸入 *pip3 install --upgrade tensorflow* 完成安裝



輸入安裝指令

實驗環境 – TensorFlow (2/2)

- 輸入指令，驗證 TensorFlow 安裝與版本

import tensorflow as tf

print(tf.__version__)

hello = tf.constant('Hello, TensorFlow!')

sess = tf.Session()

print(sess.run(hello))

- 執行結果

1.6.0

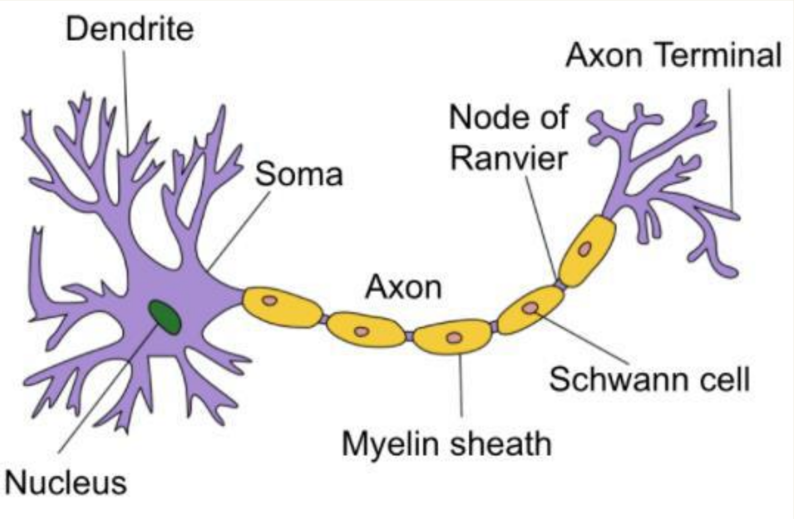
b'Hello, TensorFlow!'

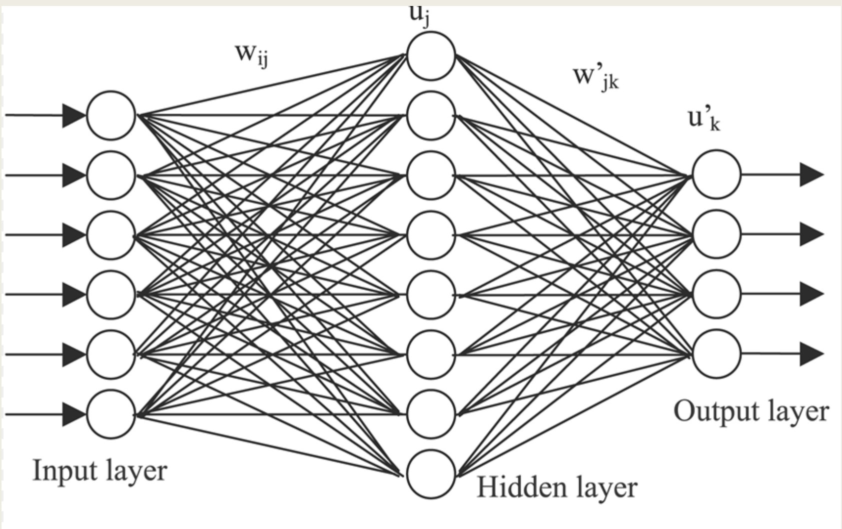
```
D:\GoogleDrive\106-1course\SOC_design\L1>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05)
[MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for
more information.
>>> import tensorflow as tf
C:\Users\Superbbman\AppData\Local\Programs\Python\Pyt
hon35\lib\site-packages\h5py\__init__.py:36: FutureWa
rning: Conversion of the second argument of issubdtyp
e from `float` to `np.floating` is deprecated. In fut
ure, it will be treated as `np.float64 == np.dtype(fl
oat).type`.
  from ._conv import register_converters as _register
_converters
>>>
>>> print(tf.__version__)
1.6.0
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
2018-03-07 18:01:49.008812: I C:\tf_jenkins\workspace
\rel-win\M\windows\PY\35\tensorflow\core\platform\cpu
_feature_guard.cc:140] Your CPU supports instructions
 that this TensorFlow binary was not compiled to use:
 AVX2
>>> print(sess.run(hello))
b'Hello, TensorFlow!'
>>>
```

驗證範例

Deep Neural Network (DNN)

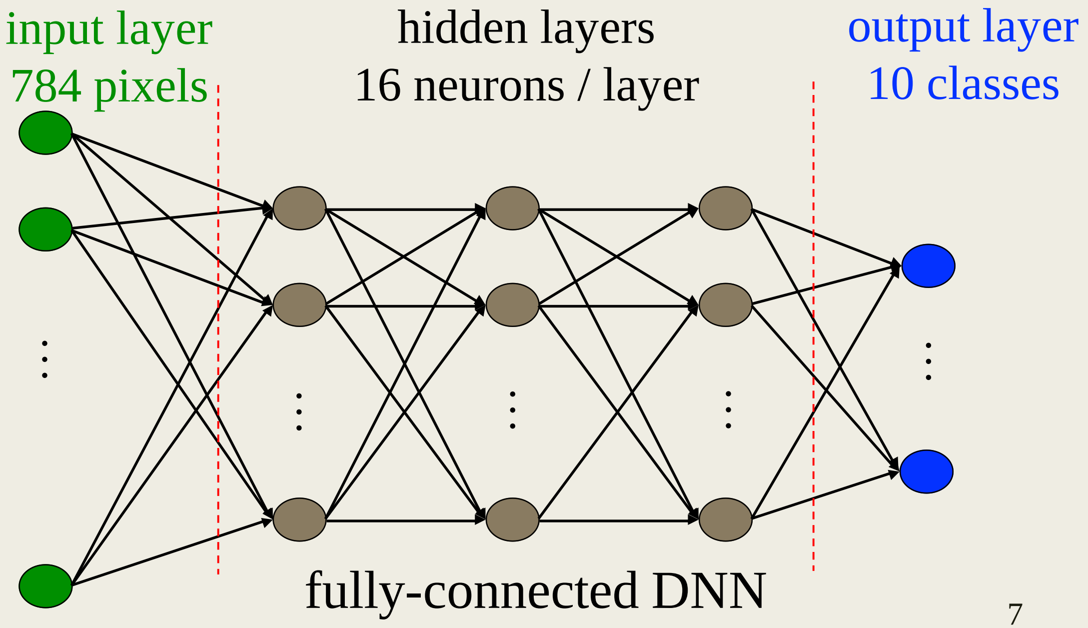
- 人工神經網路，是一種模仿生物神經網路的數學模型，通過統計學的方法，人工神經網路能夠類似人具有簡單、快速地的判斷能力，比起正式的邏輯學推理演算更具有優勢。
- 神經網路如同其他機器學習方法被廣泛的運用，例如機器的影像辨識和語音識別，以及知名的 AlphaGo 等。





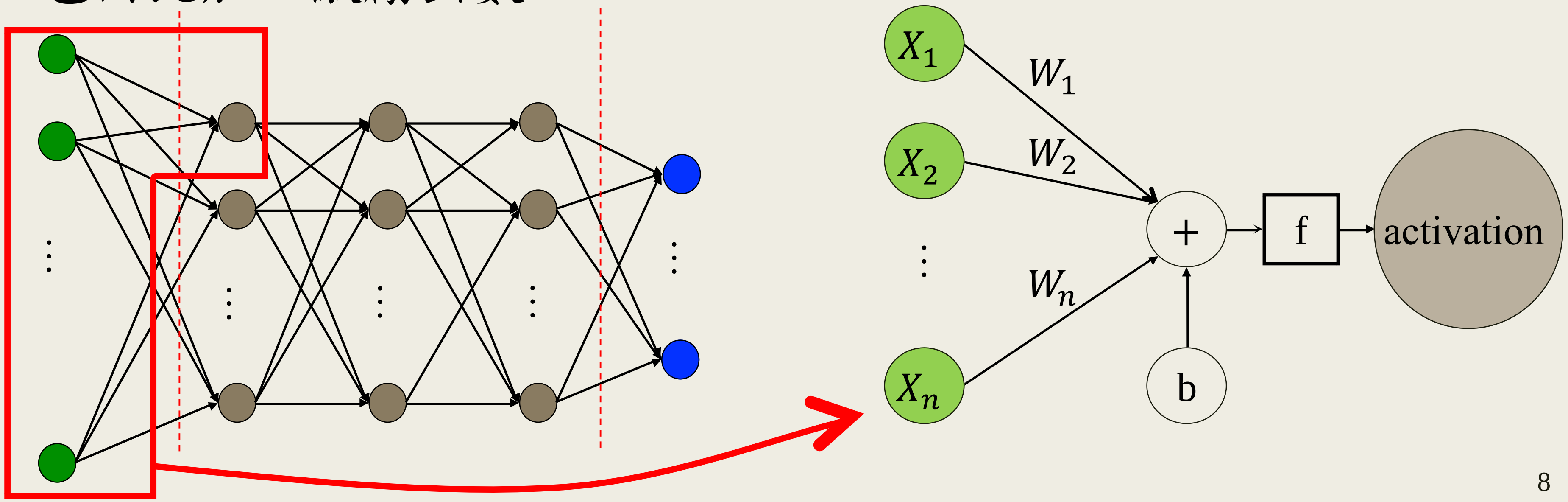
DNN架構

- 依照層級可以將 DNN 分為輸入層、隱藏層與輸出層三部分
- 輸入層的各個神經元既是 DNN 輸入，也是資料的特徵
- 隱藏層為輸出與輸入層之間的部分，可以根據情況有不同的數量
- 輸出層通常在回歸問題與二元問題時只有單個神經元，但在多元分類時會出現多個神經元



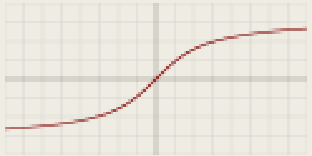
神經元計算

- DNN 的各神經元皆代表數值，下圖可見第一層神經元 X_i 有對應的權重 W_i，對這些輸入的加權總合加上偏置 b，再代入激勵函數可以得到第二層的神經元
- 將上述套用在其他的神經元，我們可以依序得到 DNN 各神經元的數值
- DNN 的神經元若僅做加權計算，則輸出輸入脫離不了線性關係，喪失神經網路的意義，也因此加入激勵函數

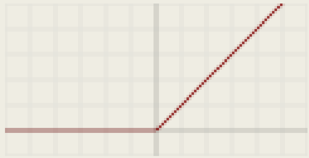


常見的激勵函數

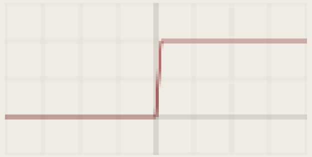
- Sigmoid：$f(x) = \frac{1}{1+e^{-x}}$



- ReLU： $f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases}$



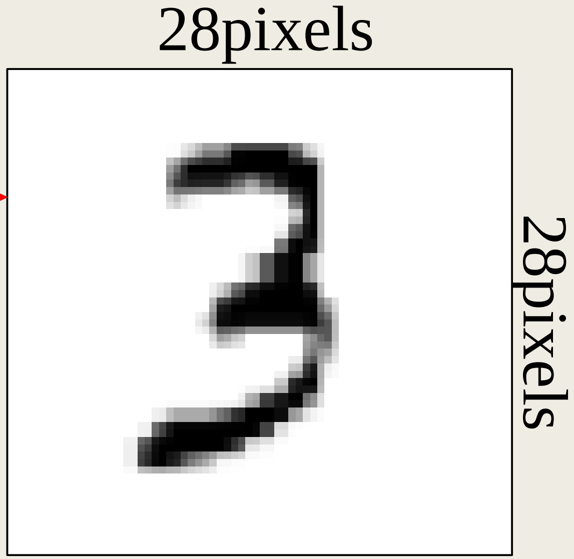
- Binary step：$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geq 0 \end{cases}$



MNIST Dataset

- MNIST資料庫由70,000筆資料構成，每一筆圖片由784個像素組成；在機器學習的領域中，一般將70,000筆資料分成訓練資料55,000筆、驗證資料5,000筆與測試資料10,000筆

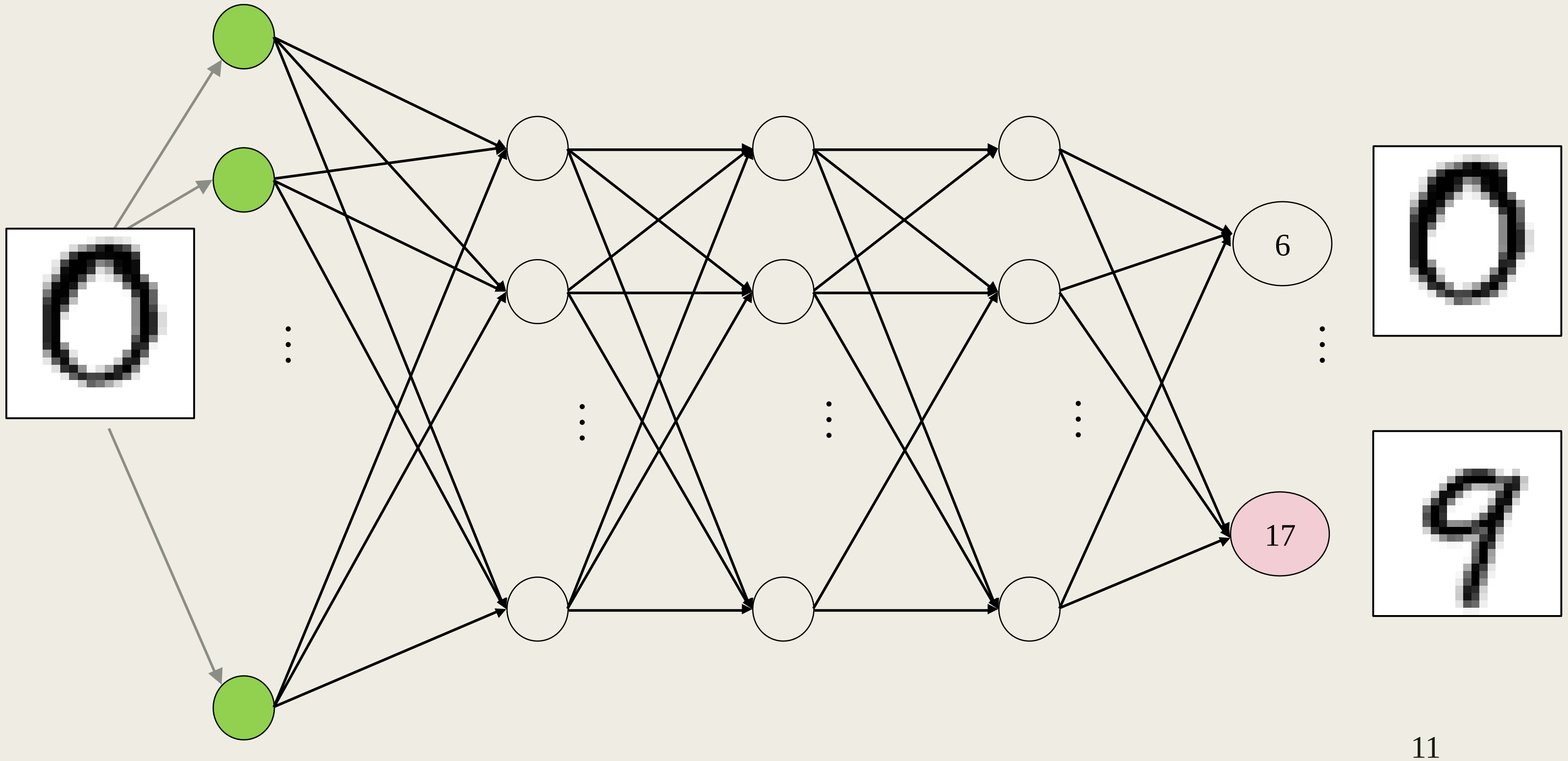
部分MNIST手寫數字



MNIST手寫數字3

DNN for MNIST

- 數值最大的輸出層神經元為 DNN 的回答
- 在DNN 還未受過訓練的情況下準確率僅約10%



範例程式一 – 手寫數字辨識 DNN

1. 開啟 dnn_example.py ，觀察程式碼與執行結果

```
  from ._conv import register_converters as _register_converters
Extracting /tmp/tensorflow/mnist/input_data\train-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data\train-labels-idx1-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data\t10k-images-idx3-ubyte.gz
Extracting /tmp/tensorflow/mnist/input_data\t10k-labels-idx1-ubyte.gz
2018-03-09 22:04:36.232557: I C:\tf_jenkins\workspace\rel-win\M\windows\PY\35\tensorflow\core\platform\cpu_feature_guard
.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Step 0: loss = 2.31 (0.109 sec)
Step 100: loss = 0.71 (0.001 sec)
Step 200: loss = 0.47 (0.001 sec)
Step 300: loss = 0.31 (0.001 sec)
Step 400: loss = 0.25 (0.001 sec)
Training Data Eval:
  Num examples: 55000  Num correct: 49590  Precision @ 1: 0.9016
Validation Data Eval:
  Num examples: 5000  Num correct: 4581  Precision @ 1: 0.9162
Test Data Eval:
  Num examples: 10000  Num correct: 9101  Precision @ 1: 0.9101

 -------------------------------------------------------
Neuron number in input  layer: 784
Neuron number in hidden layer: 16
Neuron number in output layer: 10
Weight number                : 12704
Bias   number                : 26
請按任意鍵繼續 . . .
```

範例程式一 – 手寫數字辨識 DNN

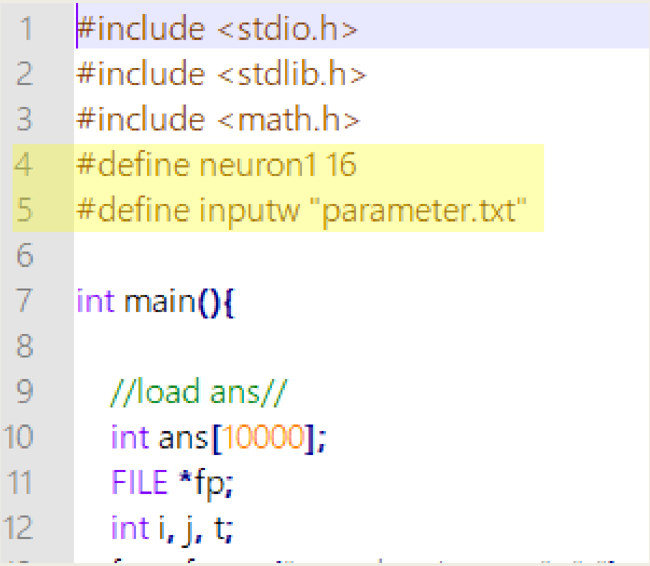
2. 修改 dnn_example.py 內的執行次數與隱藏層神經元數可以改善準確率

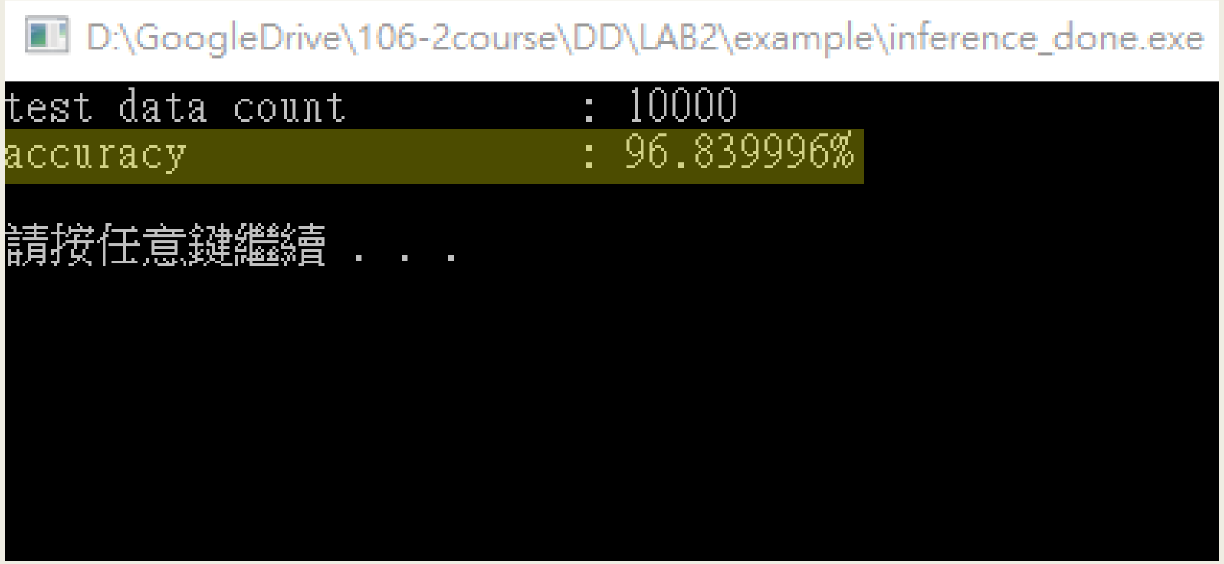
```
if __name__ == '__main__':
  parser = argparse.ArgumentParser()
  parser.add_argument(
      '--learning_rate',
      type=float,
      default=0.15,
      help='Initial learning rate.'
  )
  parser.add_argument(
      '--max_steps',
      type=int,
      default=5000,
      help='Number of steps to run trainer.'
  )
  parser.add_argument(
      '--hidden1',
      type=int,
      default=32,
      help='Number of units in hidden layer 1.'
  )

  parser.add_argument(
      '--batch_size',
      type=int,
      default=100,
      help='Batch size.  Must divide evenly into
  )
```

```
Step 2000: loss = 0.13 (0.017 sec)
Step 2100: loss = 0.05 (0.002 sec)
Step 2200: loss = 0.15 (0.093 sec)
Step 2300: loss = 0.15 (0.002 sec)
Step 2400: loss = 0.40 (0.001 sec)
Step 2500: loss = 0.17 (0.001 sec)
Step 2600: loss = 0.18 (0.001 sec)
Step 2700: loss = 0.21 (0.001 sec)
Step 2800: loss = 0.15 (0.001 sec)
Step 2900: loss = 0.17 (0.001 sec)
Training Data Eval:
  Num examples: 55000  Num correct: 52391  Precision @ 1: 0.9526
Validation Data Eval:
  Num examples: 5000  Num correct: 4770  Precision @ 1: 0.9540
Test Data Eval:
  Num examples: 10000  Num correct: 9530  Precision @ 1: 0.9530

 -----------------------------------------------------------
Neuron number in input  layer: 784
Neuron number in hidden layer: 32
Neuron number in output layer: 10
Weight number                : 25408
Bias   number                : 42
請按任意鍵繼續 . . .
```

範例程式二 － C Implementation

1. 修改inference.c內的隱藏層神經元數，使其對應修改後的DNN架構
2. 確認inference.c能夠讀取dnn_example.py輸出的parameter.txt
3. 執行inference.c得到的準確率與原本DNN執行後的結果幾乎相同





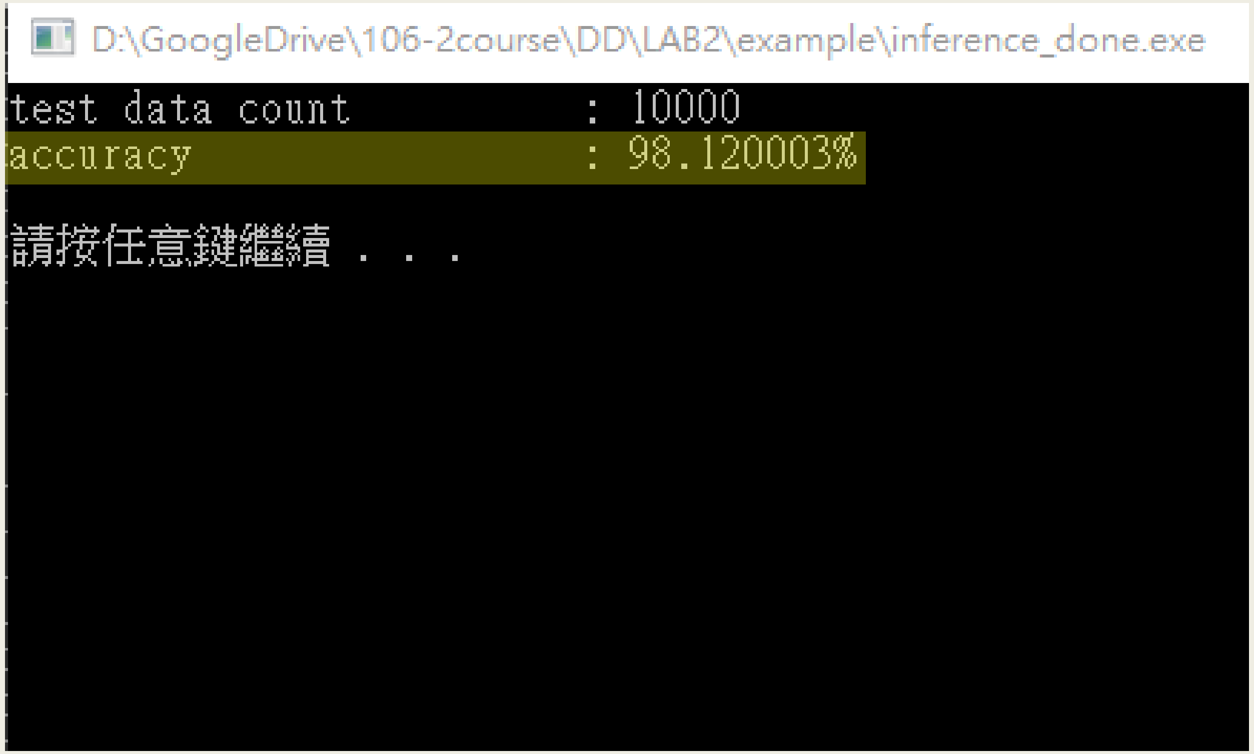
作業 – Part 1 DNN 架構

- 題目：修改範例一 DNN 架構，使準確率超過97%

```
Training Data Eval:
  Num examples: 55000  Num correct: 54979  Precision @ 1: 0.9996
Validation Data Eval:
  Num examples: 5000  Num correct: 4905  Precision @ 1: 0.9810
Test Data Eval:
  Num examples: 10000  Num correct: 9811  Precision @ 1: 0.9811
Step 29000: loss = 0.01 (0.017 sec)
Step 29100: loss = 0.03 (0.002 sec)
Step 29200: loss = 0.01 (0.002 sec)
Step 29300: loss = 0.01 (0.002 sec)
Step 29400: loss = 0.00 (0.002 sec)
Step 29500: loss = 0.00 (0.002 sec)
Step 29600: loss = 0.00 (0.002 sec)
Step 29700: loss = 0.01 (0.095 sec)
Step 29800: loss = 0.00 (0.001 sec)
Step 29900: loss = 0.01 (0.001 sec)
Training Data Eval:
  Num examples: 55000  Num correct: 54983  Precision @ 1: 0.9997
Validation Data Eval:
  Num examples: 5000  Num correct: 4909  Precision @ 1: 0.9818
Test Data Eval:
  Num examples: 10000  Num correct: 9812  Precision @ 1: 0.9812
```

作業 – Part 2 DNN in C

■ 題目：修改範例二，使準確率與作業一一致



作業 – Part 3 手寫數字辨識程式

- 題目：結合 Lab1 與 Lab2 的作業二，做出手寫數字辨識程式
  - 功能一：具有 Lab1 進階練習的三個功能
  - 功能二：可由按鈕執行 DNN，顯示辨識結果在新圖框
  - 功能三：執行 DNN 後，以直方圖顯示輸出層數值

課程評分

- Demo 時間：四梯次時間分別為 19:30、19:50、 20:10 與 20:30
- Demo 梯次：與 Lab 1 相同
- Demo 地點：計中217
- 評分方式：Part 1/2/3各佔20分，且依 DNN 權重數做排名給分， 數量越少排名則越前面，該部分佔40分，給分規則如表格

```
Training Data Eval:
  Num examples: 55000  Num correct: 49590  Precision @ 1: 0.9016
Validation Data Eval:
  Num examples: 5000  Num correct: 4581  Precision @ 1: 0.9162
Test Data Eval:
  Num examples: 10000  Num correct: 9101  Precision @ 1: 0.9101

 -------------------------------------------------------
Neuron number in input  layer: 784
Neuron number in hidden layer: 16
Neuron number in output layer: 10
Weight number                : 12704    DNN 權重數
Bias   number                : 26
請按任意鍵繼續 . . .
```

排名	分數
1	40
2	30
3 ~ 10	20
11 ~ 30	10
其餘	0

排名給分表