1082 – Deep Learning

LabHomework 2 – TensorFlow 與 MNIST

授課老師：朱守禮

學生組別：第八組

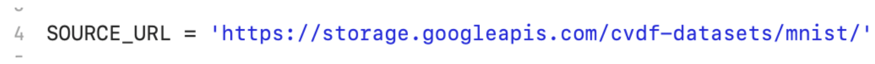
學號：10526150／10527105／10627122／10627146

姓名：葉洧綾／周利亞／陳俐欣／陳浩瑋

1. 實驗操作步驟

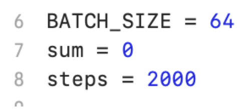
開啟 Lab1 所架設好之 VMware，藉由 firework 開啟 ilearning，下載本次實驗所需的程式範例包，將範例包解壓縮，開啟 terminal，切換至範例程式所在之目錄，藉由“python3 Mnist_DNN.py”執行程式，等待程式訓練結果，查看 accuracy；嘗試藉由調整 batchsize、steps 及層數，來提高 accuracy 至 0.9。

2. 程式碼說明



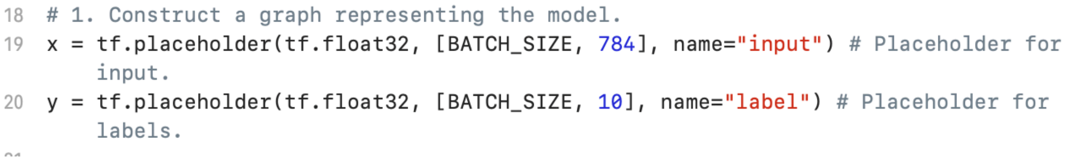
```
4  SOURCE_URL = 'https://storage.googleapis.com/cvdf-datasets/mnist/'
```

下載 Mnist 的 dataset。



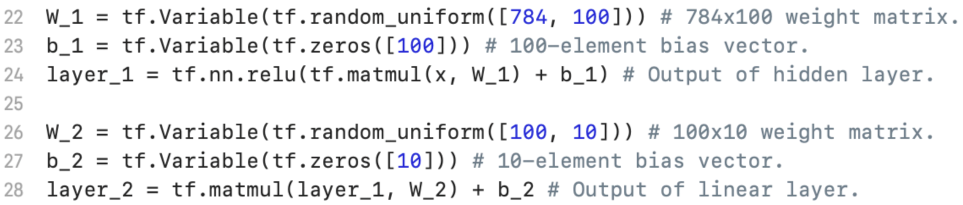
```
6  BATCH_SIZE = 64
7  sum = 0
8  steps = 2000
```

Batch_size 及訓練時的步數 steps。



```
18  # 1. Construct a graph representing the model.
19  x = tf.placeholder(tf.float32, [BATCH_SIZE, 784], name="input") # Placeholder for
        input.
20  y = tf.placeholder(tf.float32, [BATCH_SIZE, 10], name="label") # Placeholder for
        labels.
```

第 19 行為讀取 dataset，一次讀取 batchsize 的張數，每張為 28*28 的像素；第 20 行為拿取 output，一樣為 batchsize 的張數，總共分成 0 到 9，共 10 種。



```
22  W_1 = tf.Variable(tf.random_uniform([784, 100])) # 784x100 weight matrix.
23  b_1 = tf.Variable(tf.zeros([100])) # 100-element bias vector.
24  layer_1 = tf.nn.relu(tf.matmul(x, W_1) + b_1) # Output of hidden layer.
25
26  W_2 = tf.Variable(tf.random_uniform([100, 10])) # 100x10 weight matrix.
27  b_2 = tf.Variable(tf.zeros([10])) # 10-element bias vector.
28  layer_2 = tf.matmul(layer_1, W_2) + b_2 # Output of linear layer.
```

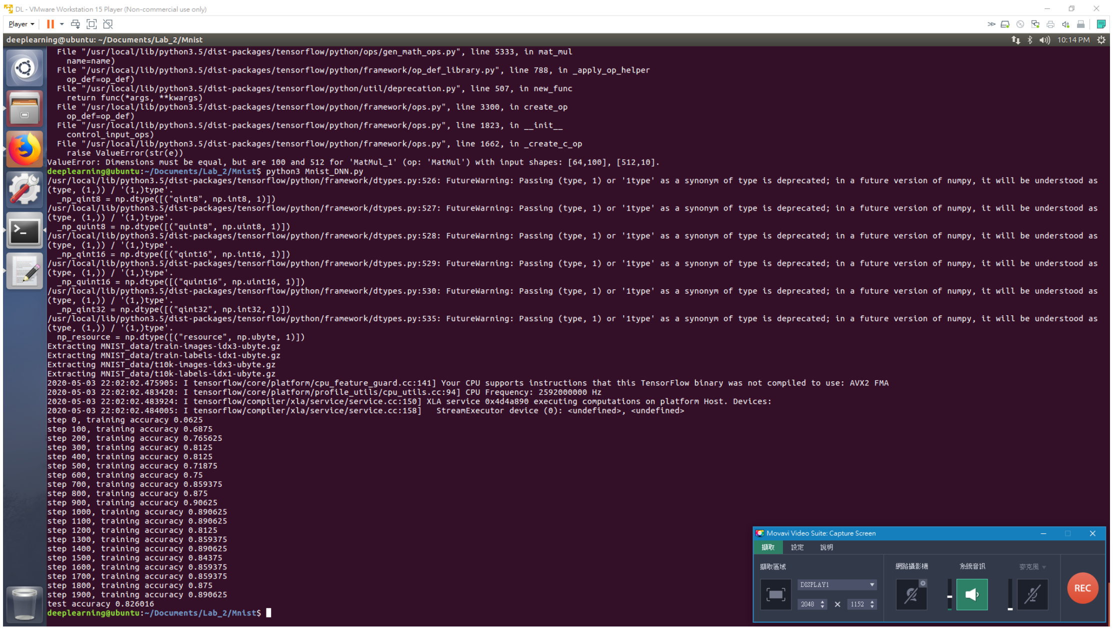
22-24 為第一層，將[batchsize,784]乘上經由 random 產生的[784,100]權重，轉換成[batchsize,100]再加上偏移量成為第一層 output。

26-28 為第二層，也是範例程式的輸出層，將[batchsize,100]乘上經由 random 產生的[100,10（輸出數量）]權重，轉換成[batchsize,10]再加上偏移量成為輸出結果。

```python
# 2. Add nodes that represent the optimization algorithm.
with tf.name_scope('Loss'):
  loss = tf.nn.softmax_cross_entropy_with_logits_v2(labels=y, logits=layer_2)
  tf.summary.scalar('loss',tf.reduce_mean(loss))
train_op = tf.train.AdagradOptimizer(0.01).minimize(loss)

with tf.name_scope('accuracy'):
  correct_prediction = tf.equal(tf.argmax(layer_2,1), tf.argmax(y,1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
  tf.summary.scalar('accuracy', accuracy)
```

比較辨識結果與實際答案，計算 loss 及 accuracy 值。

3. 執行結果

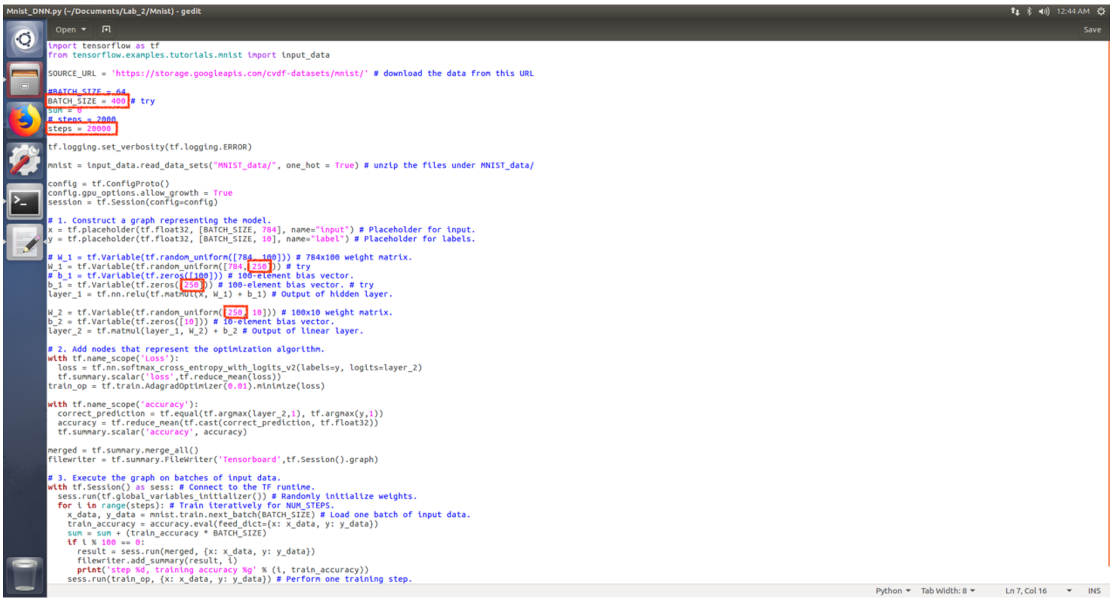

DL - VMware Workstation 15 Player (Non-commercial use only)
Player
deeplearning@ubuntu: ~/Documents/Lab_2/Mnist
10:14 PM
Movavi Video Suite: Capture Screen
擷取
設定
說明
擷取區域
DISPLAY1
2048
1152
網路攝影機
系統音訊
麥克風
REC

```
  File "/usr/local/lib/python3.5/dist-packages/tensorflow/python/ops/gen_math_ops.py", line 5333, in mat_mul
    name=name)
  File "/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/op_def_library.py", line 788, in _apply_op_helper
    op_def=op_def)
  File "/usr/local/lib/python3.5/dist-packages/tensorflow/python/util/deprecation.py", line 507, in new_func
    return func(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/ops.py", line 3300, in create_op
    op_def=op_def)
  File "/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/ops.py", line 1823, in __init__
    control_input_ops)
  File "/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/ops.py", line 1662, in _create_c_op
    raise ValueError(str(e))
ValueError: Dimensions must be equal, but are 100 and 512 for 'MatMul_1' (op: 'MatMul') with input shapes: [64,100], [512,10].
deeplearning@ubuntu:~/Documents/Lab_2/Mnist$ python3 Mnist_DNN.py
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dtypes.py:527: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dtypes.py:528: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dtypes.py:529: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dtypes.py:530: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/lib/python3.5/dist-packages/tensorflow/python/framework/dtypes.py:535: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
2020-05-03 22:02:02.475905: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
2020-05-03 22:02:02.483420: I tensorflow/core/platform/profile_utils/cpu_utils.cc:94] CPU Frequency: 2592000000 Hz
2020-05-03 22:02:02.483924: I tensorflow/compiler/xla/service/service.cc:150] XLA service 0x4d4a890 executing computations on platform Host. Devices:
2020-05-03 22:02:02.484005: I tensorflow/compiler/xla/service/service.cc:158]   StreamExecutor device (0): <undefined>, <undefined>
step 0, training accuracy 0.0625
step 100, training accuracy 0.6875
step 200, training accuracy 0.765625
step 300, training accuracy 0.8125
step 400, training accuracy 0.8125
step 500, training accuracy 0.71875
step 600, training accuracy 0.75
step 700, training accuracy 0.859375
step 800, training accuracy 0.875
step 900, training accuracy 0.90625
step 1000, training accuracy 0.890625
step 1100, training accuracy 0.890625
step 1200, training accuracy 0.8125
step 1300, training accuracy 0.859375
step 1400, training accuracy 0.890625
step 1500, training accuracy 0.84375
step 1600, training accuracy 0.859375
step 1700, training accuracy 0.859375
step 1800, training accuracy 0.875
step 1900, training accuracy 0.890625
test accuracy 0.826016
deeplearning@ubuntu:~/Documents/Lab_2/Mnist$
```

4. 遇到的問題(or 解決方法)

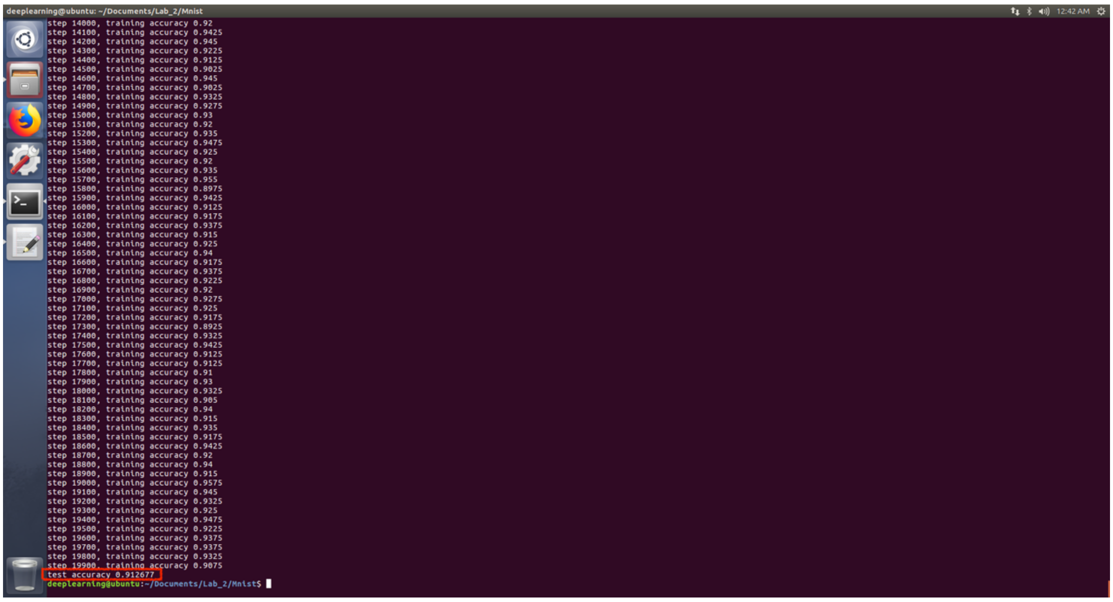
提高辨識精確度，藉由修改 batch_size、steps 及神經網路的深度及各層分類數量，嘗試提高 accuracy。

5. 以上內容請附圖並說明



```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

SOURCE_URL = 'https://storage.googleapis.com/cvdf-datasets/mnist/' # download the data from this URL

#BATCH_SIZE = 64
BATCH_SIZE = 400 # try
sum = 0
# steps = 2000
steps = 20000

tf.logging.set_verbosity(tf.logging.ERROR)

mnist = input_data.read_data_sets("MNIST_data/", one_hot = True) # unzip the files under MNIST_data/

config = tf.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.Session(config=config)

# 1. Construct a graph representing the model.
x = tf.placeholder(tf.float32, [BATCH_SIZE, 784], name="input") # Placeholder for input.
y = tf.placeholder(tf.float32, [BATCH_SIZE, 10], name="label") # Placeholder for labels.

# W_1 = tf.Variable(tf.random_uniform([784, 100])) # 784x100 weight matrix.
W_1 = tf.Variable(tf.random_uniform([784, 250])) # try
# b_1 = tf.Variable(tf.zeros([100])) # 100-element bias vector.
b_1 = tf.Variable(tf.zeros([250])) # 100-element bias vector. # try
layer_1 = tf.nn.relu(tf.matmul(x, W_1) + b_1) # Output of hidden layer.

W_2 = tf.Variable(tf.random_uniform([250, 10])) # 100x10 weight matrix.
b_2 = tf.Variable(tf.zeros([10])) # 10-element bias vector.
layer_2 = tf.matmul(layer_1, W_2) + b_2 # Output of linear layer.

# 2. Add nodes that represent the optimization algorithm.
with tf.name_scope('Loss'):
  loss = tf.nn.softmax_cross_entropy_with_logits_v2(labels=y, logits=layer_2)
  tf.summary.scalar('loss',tf.reduce_mean(loss))
train_op = tf.train.AdagradOptimizer(0.01).minimize(loss)

with tf.name_scope('accuracy'):
  correct_prediction = tf.equal(tf.argmax(layer_2,1), tf.argmax(y,1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
  tf.summary.scalar('accuracy', accuracy)

merged = tf.summary.merge_all()
filewriter = tf.summary.FileWriter('Tensorboard',tf.Session().graph)

# 3. Execute the graph on batches of input data.
with tf.Session() as sess: # Connect to the TF runtime.
  sess.run(tf.global_variables_initializer()) # Randomly initialize weights.
  for i in range(steps): # Train iteratively for NUM_STEPS.
    x_data, y_data = mnist.train.next_batch(BATCH_SIZE) # Load one batch of input data.
    train_accuracy = accuracy.eval(feed_dict={x: x_data, y: y_data})
    sum = sum + (train_accuracy * BATCH_SIZE)
    if i % 100 == 0:
      result = sess.run(merged, {x: x_data, y: y_data})
      filewriter.add_summary(result, i)
      print('step %d, training accuracy %g' % (i, train_accuracy))
    sess.run(train_op, {x: x_data, y: y_data}) # Perform one training step.
```

在嘗試的過程中發現當 batch_size 增加，每一步所需要花費的時間就越多，因此必須要找到最適合的 batch_size，讓結果和耗時都能令人接受。起初將 batch_size 固定在 128，透過調整 step 的大小和層與層間的輸出數量來改變精準度，但嘗試多次後發現再怎麼提高 step 的階數精準度都還是維持在 0.87 左右，於是慢慢增加 batch_size 的大小，發現當只調整 step，遇到瓶頸時，適時改變 batch_size 可以有更大的機會提高精準度。而調整層與層間的輸出數量是可以有限度的提升精準度，但當數字改為 400 左右時會發現儘管再提高數值，精準度也不會再提升，反而會增加每一步的耗時。



```
step 14000, training accuracy 0.92
step 14100, training accuracy 0.9425
step 14200, training accuracy 0.945
step 14300, training accuracy 0.9225
step 14400, training accuracy 0.9125
step 14500, training accuracy 0.9025
step 14600, training accuracy 0.945
step 14700, training accuracy 0.9025
step 14800, training accuracy 0.9325
step 14900, training accuracy 0.9275
step 15000, training accuracy 0.93
step 15100, training accuracy 0.92
step 15200, training accuracy 0.935
step 15300, training accuracy 0.9475
step 15400, training accuracy 0.925
step 15500, training accuracy 0.92
step 15600, training accuracy 0.935
step 15700, training accuracy 0.955
step 15800, training accuracy 0.8975
step 15900, training accuracy 0.9425
step 16000, training accuracy 0.9125
step 16100, training accuracy 0.9175
step 16200, training accuracy 0.9375
step 16300, training accuracy 0.915
step 16400, training accuracy 0.925
step 16500, training accuracy 0.94
step 16600, training accuracy 0.9175
step 16700, training accuracy 0.9375
step 16800, training accuracy 0.9225
step 16900, training accuracy 0.92
step 17000, training accuracy 0.9275
step 17100, training accuracy 0.925
step 17200, training accuracy 0.9175
step 17300, training accuracy 0.8925
step 17400, training accuracy 0.9325
step 17500, training accuracy 0.9425
step 17600, training accuracy 0.9125
step 17700, training accuracy 0.9125
step 17800, training accuracy 0.91
step 17900, training accuracy 0.93
step 18000, training accuracy 0.9325
step 18100, training accuracy 0.905
step 18200, training accuracy 0.94
step 18300, training accuracy 0.915
step 18400, training accuracy 0.935
step 18500, training accuracy 0.9175
step 18600, training accuracy 0.9425
step 18700, training accuracy 0.92
step 18800, training accuracy 0.94
step 18900, training accuracy 0.915
step 19000, training accuracy 0.9575
step 19100, training accuracy 0.945
step 19200, training accuracy 0.9325
step 19300, training accuracy 0.925
step 19400, training accuracy 0.9475
step 19500, training accuracy 0.9225
step 19600, training accuracy 0.9375
step 19700, training accuracy 0.9375
step 19800, training accuracy 0.9325
step 19900, training accuracy 0.9075
test accuracy 0.912677
deeplearning@ubuntu:~/Documents/Lab_2/Mnist$ 
```

當 batch_size 設為 400，step 設為 200000，層與層間輸出的數量設為 250 時能得到 0.912677 的精準度。

6. 心得

<u>葉洧綾：</u>

<u>周利亞：</u>

<u>陳俐欣：</u>

這次的 Lab 讓人能夠更了解整個 DNN 的運作，儘管還無法親自刻出一個 DNN 模型，但我終於知道要如何針對一個模型去做調整，修改模型的各種參數，包括 batch_size、step 等，透過不同的數字組合，來改變一個模型的精準度。透過這次的練習，不僅更理解層與層間是如何連結的，也更親身體會參數間的奧妙，精準度和參數並不一定是呈線性變化，因此若只專注於提高其中一個參數時，

到某個階段就會面臨無法再提高的瓶頸，也因此明白一個深度學習模型不會有一個正確解答，只會因為不同的參數組合而有更好的答案。

陳浩瑋：

這是第一次可以看懂 DNN 的運作過程，而且可以實際調整其中的過程，十分的有趣：在一開始的時候只想要增加 steps 來提高準確度，但發現調整到一定的 steps 之後，雖然還是有提高的準確度，但是都只是一些些而已，而且我的電腦已經蠻老的，所以也跑必較久，所以後來想說要來改訓練的層數深度、參數，但是萬萬沒想到反而越改越糟，準確度移植在下降，最低只剩下 0.1 多，看來調整深度、參數，不是輕易就可以調整出來的東西。