Introduction to TensorFlow

Convolutional Neural Networks with TensorFlow

郭耀仁

大綱

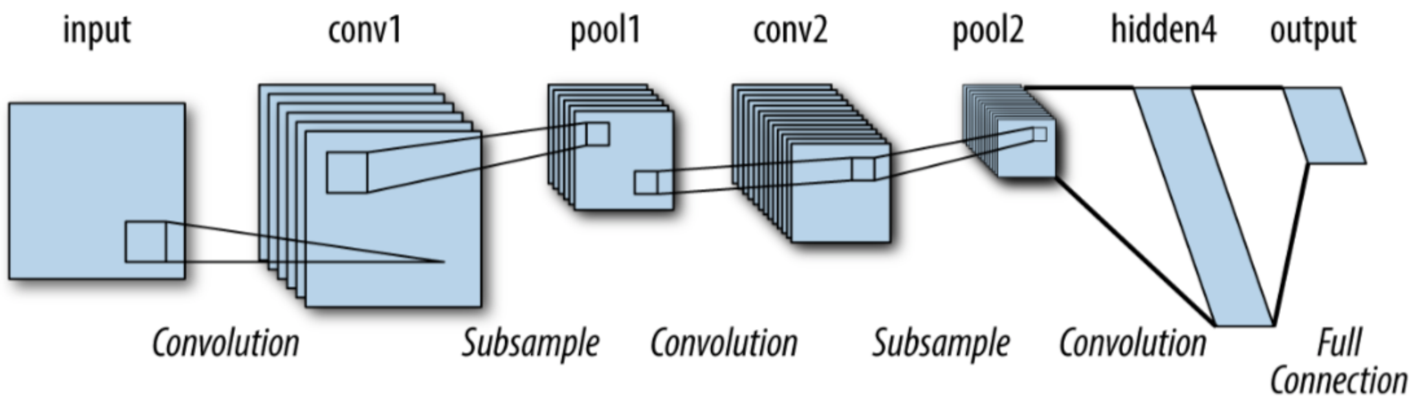
- 關於 Convolutional Neural Networks
- 取得資料
- 建構 TensorFlow 計算圖形
- 訓練
- 隨堂練習

關於 Convolutional Neural Networks

有別於 Fully Connected Deep Network

Convolutional Neural Networks(CNN) 並非只有單純的 Input、Hidden、Output layers

標準的 Convolutional Neural Networks 結構

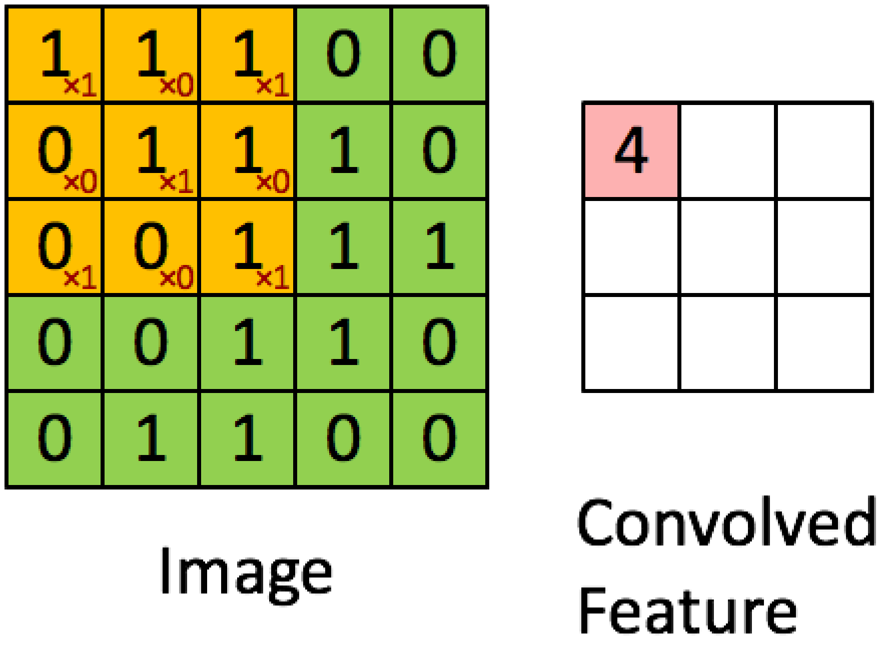


Source: [TensorFlow for Deep Learning (https://www.amazon.com/TensorFlow-Deep-Learning-Regression-Reinforcement/dp/1491980451)](https://www.amazon.com/TensorFlow-Deep-Learning-Regression-Reinforcement/dp/1491980451)

各司其職的 Layers

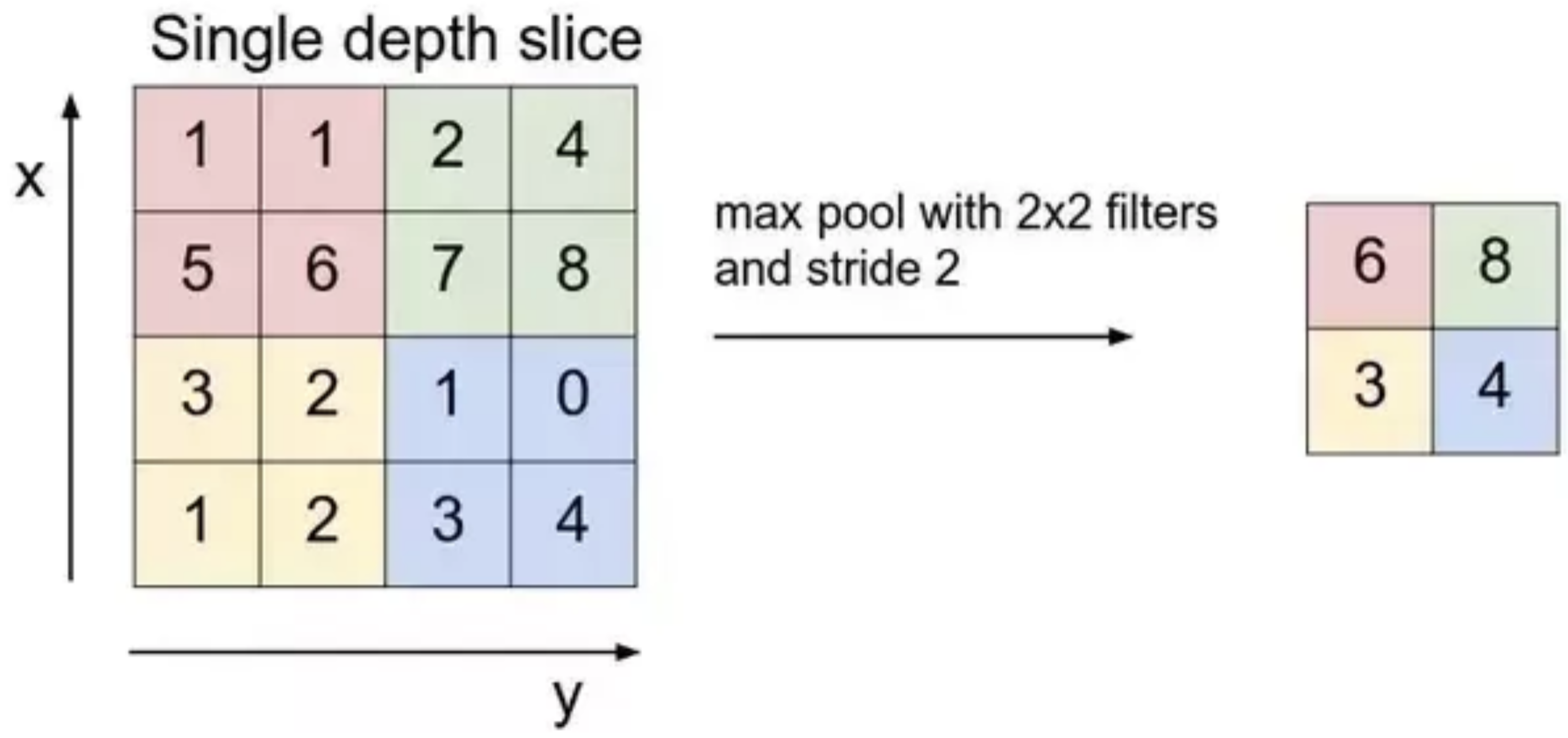
- 輸入層
- 卷積層（Convolution Layer）： 透過套用濾鏡矩陣提取圖片特徵
- 池化層 (Pooling Layer)：減少需要訓練的參數、並確定提取的特徵能夠被保留下來
- 平坦層（Flatten Layer）： 為最後輸入多層感知器準備
- 多層感知器（Fully-Connected Deep Network）
- 輸出層

卷積層示意



Source: Artificial Inteligence (https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolution.html)

池化層示意



Source: [Quora (https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks)](https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks)

取得資料

In [1]:
```
import pandas as pd

train_url = "https://s3-ap-northeast-1.amazonaws.com/kaggle-getting-started/mnist/
train.csv"
train = pd.read_csv(train_url)
X_arr = train.values[:, 1:]
y_arr = train.values[:, 0]
```

印出來看看

```
In [2]: import matplotlib.pyplot as plt

        first_img = X_arr[0, :].reshape(28, 28)
        plt.matshow(first_img, cmap = plt.get_cmap('gray'))
        plt.show()
```

```
<Figure size 480x480 with 1 Axes>
```

切割資料

```
In [3]: from sklearn.model_selection import train_test_split

        X_train, X_test, y_train, y_test = train_test_split(X_arr, y_arr, test_size=0.3, r
        andom_state=123)
        X_train = X_train[:700, :]
        y_train = y_train[:300]
        print(X_train.shape)
        print(X_test.shape)
        print(y_train.shape)
        print(y_test.shape)
```

```
(700, 784)
(12600, 784)
(300,)
(12600,)
```

建構 TensorFlow 計算圖形

輸入層：解析度 28x28 的手寫數字圖片（784 個神經元）

In [4]:
```
import tensorflow as tf
import numpy as np

n_features = X_train.shape[1]
n_labels = np.unique(y_train).size
with tf.name_scope('input-layer'):
  x = tf.placeholder(tf.float32, shape=(None, n_features))
  y = tf.placeholder(tf.float32, shape=(None, n_labels))
```

自訂初始化權重的函數

In [5]:
```
def weight_variable(shape):
  initial = tf.truncated_normal(stddev=0.1, shape=shape)
  return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)
```

自訂卷積層與池化層的生成函數

In [6]:
```
def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAM
E')
```

卷積神經網絡架構

- 輸入層：解析度 28x28 的手寫數字圖片（784 個神經元）
- 第一層是 Convolution 層（32 個神經元），會利用解析度 5x5 的濾鏡取出 32 個特徵，然後將圖片降維成解析度 14x14
- 第二層是 Convolution 層（64 個神經元），會利用解析度 5x5 的濾鏡取出 64 個特徵，然後將圖片降維成解析度 7x7
- 第三層是平坦層（1024 個神經元），會將圖片的 1024 個特徵攤平
- 輸出結果之前使用 Dropout 函數避免過度配適
- 第四層是輸出層（10 個神經元），使用 Softmax 函數

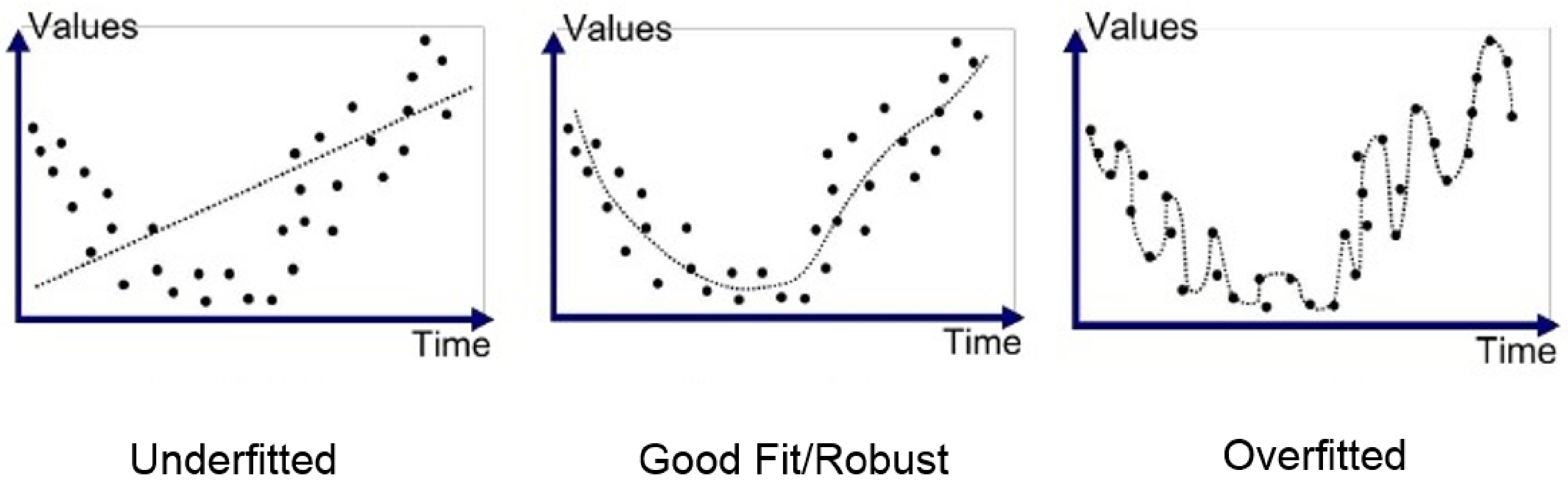
In [7]:

```
# 第一層是 Convolution 層（32 個神經元），會利用解析度 5x5 的 filter 取出 32 個特徵，然後將
圖片降維成解析度 14x14
with tf.name_scope('first-convolution-layer'):
  W_conv1 = weight_variable([5, 5, 1, 32])
  b_conv1 = bias_variable([32])
  x_image = tf.reshape(x, [-1, 28, 28, 1])
  h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
  h_pool1 = max_pool_2x2(h_conv1)
```

In [8]:
```
# 第二層是 Convolution 層 (64 個神經元) , 會利用解析度 5x5 的 filter 取出 64 個特徵, 然後將
圖片降維成解析度 7x7
with tf.name_scope('second-convolution-layer'):
  W_conv2 = weight_variable([5, 5, 32, 64])
  b_conv2 = bias_variable([64])
  h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
  h_pool2 = max_pool_2x2(h_conv2)
```

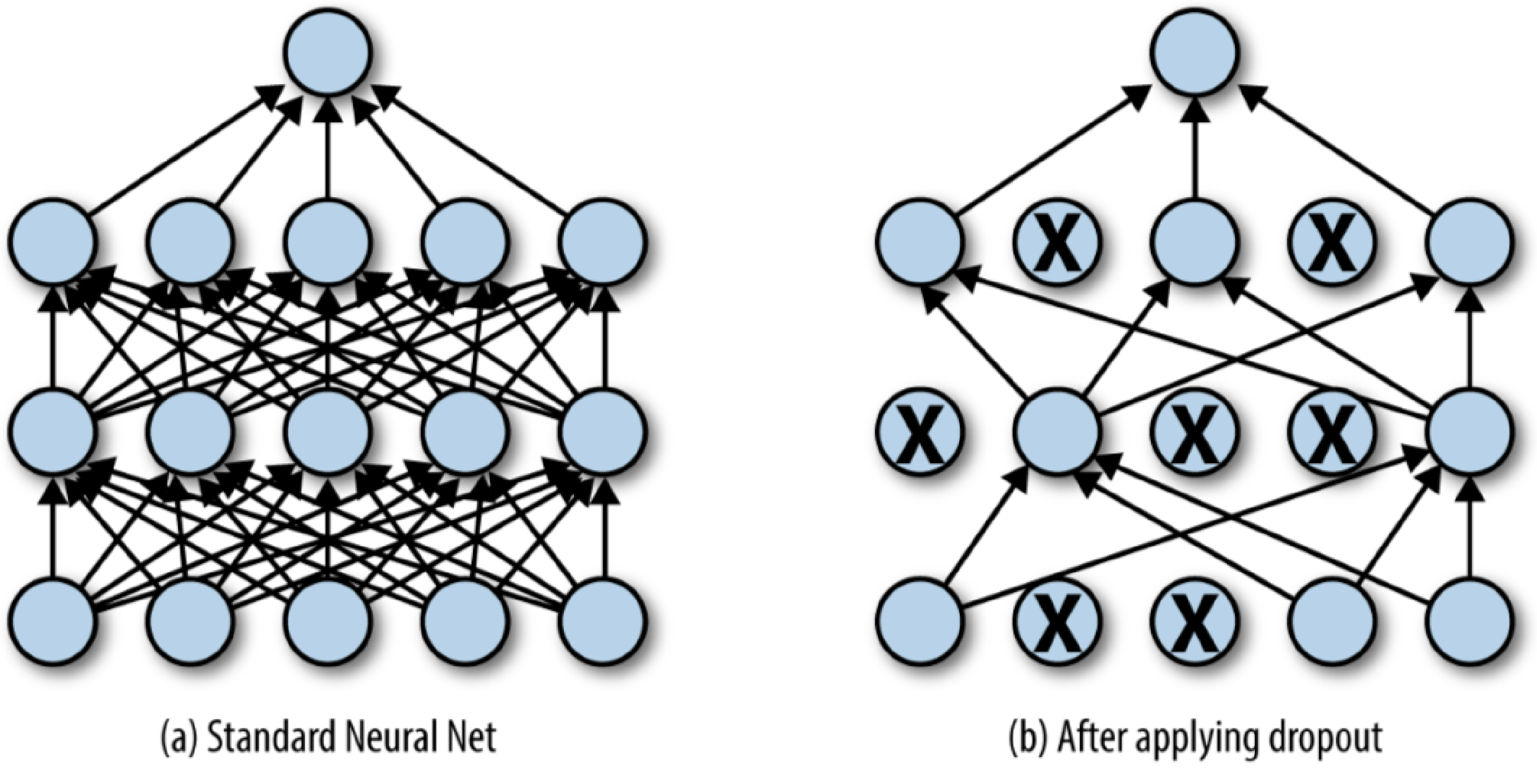
In [9]:
```
# 第三層是平坦層（1024 個神經元），會將圖片的 1024 個特徵攤平
with tf.name_scope('flatten-layer'):
  W_fc1 = weight_variable([7 * 7 * 64, 1024])
  b_fc1 = bias_variable([1024])
  h_pool2_flat = tf.reshape(h_pool2, (-1, 7 * 7 * 64))
  h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)
```

機器學習會產生配適問題



Source: [What is underfitting and overfitting in machine learning and how to deal with it (https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76)](https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76)

一般機器學習利用正規化（regularization）避免過度配適，神經網路則使用 Dropout



Source: TensorFlow for Deep Learning (https://www.amazon.com/TensorFlow-Deep-Learning-Regression-Reinforcement/dp/1491980451)

In [10]:

```
# 輸出結果之前使用 Dropout 函數避免過度配適
with tf.name_scope('dropout'):
  keep_prob = tf.placeholder(tf.float32)
  h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
```

Softmax 函數會將 10 個分類的機率最大者回傳為辨識結果

```
In [11]: # 第四層是輸出層（10 個神經元），使用 Softmax 函數輸出結果
         with tf.name_scope('output-layer'):
           W_fc2 = weight_variable([1024, 10])
           b_fc2 = bias_variable([10])
           y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2
```

寫下成本函數的公式

In [12]:
```
with tf.name_scope('loss'):
  entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=y_con
v, labels=y))
  loss = tf.reduce_sum(entropy)
```

宣告 Optimizer 與學習速率

In [13]:
```
learning_rate = 0.0001
with tf.name_scope("optimizer"):
  opt = tf.train.AdamOptimizer(learning_rate)
  optimizer = opt.minimize(loss)
```

建構 TensorFlow 計算圖形完整程式碼

```
In [15]: import tensorflow as tf
         import numpy as np

         def weight_variable(shape):
           initial = tf.truncated_normal(stddev=0.1, shape=shape)
           return tf.Variable(initial)

         def bias_variable(shape):
           initial = tf.constant(0.1, shape=shape)
           return tf.Variable(initial)

         def conv2d(x, W):
           return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

         def max_pool_2x2(x):
           return tf.nn.max_pool(x, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAM
         E')

         tf.reset_default_graph()
         n_features = X_train.shape[1]
         n_labels = np.unique(y_train).size
         learning_rate = 0.0001
         with tf.name_scope('input-layer'):
           x = tf.placeholder(tf.float32, shape=(None, n_features))
           y = tf.placeholder(tf.float32, shape=(None, n_labels))
         with tf.name_scope('first-convolution-layer'):
           W_conv1 = weight_variable([5, 5, 1, 32])
           b_conv1 = bias_variable([32])
           x_image = tf.reshape(x, [-1, 28, 28, 1])
           h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
           h_pool1 = max_pool_2x2(h_conv1)
         with tf.name_scope('second-convolution-layer'):
           W_conv2 = weight_variable([5, 5, 32, 64])
           b_conv2 = bias_variable([64])
           h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
           h_pool2 = max_pool_2x2(h_conv2)
         with tf.name_scope('flatten-layer'):
```

訓練

In [ ]:

```
n_steps = 20000
file_writer_path = "./graphs/convolutional-neural-networks"
loss_history = []
step = 0
batch_size = 50
n_obs = X_train.shape[0]

with tf.Session() as sess:
  sess.run(tf.global_variables_initializer()) # 初始化所有的變數張量!
  train_writer = tf.summary.FileWriter(file_writer_path, tf.get_default_graph())
  for i in range(n_steps):
    pos = 0
    while pos < n_obs:
      batch_X = X_train[pos:(pos + batch_size)]
      batch_y = y_train[pos:(pos + batch_size)]
    feed_dict = {
      x: batch_X,
      y: batch_y,
      keep_prob: 1.0
    }
    _, loss_ = sess.run([optimizer, loss], feed_dict=feed_dict)
    loss_history.append(loss_)
    if i % 1000 == 0:
      print("step {}, loss: {}".format(i, loss_))
    step += 1
    pos += batch_size
```

隨堂練習

使用 sklearn 的 digits 資料集並利用 TensorFlow 建立一個 Convolutional Neural Network Classifier

In [16]:
```python
from sklearn.datasets import load_digits

digits = load_digits()
print(digits.DESCR)
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 5620
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digi
ts

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of

4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
```

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
1994.

.. topic:: References

- C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition, MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University.
- E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
- Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin. Linear dimensionalityreduction using relevance weighted LDA. School of Electrical and Electronic Engineering Nanyang Technological University. 2005.
- Claudio Gentile. A New Approximate Maximal Margin Classification Algorithm. NIPS. 2000.

```
In [17]: X_arr = digits.data
         y_arr = digits.target
         print(X_arr.shape)
         print(y_arr.shape)
```

```
(1797, 64)
(1797,)
```