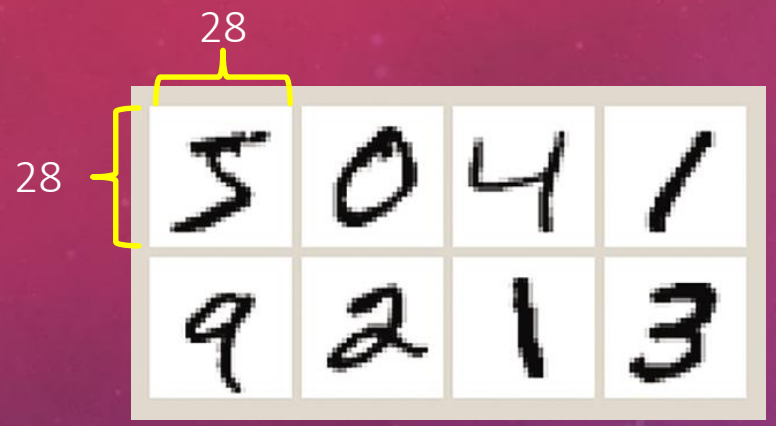
多 媒 體 系 統
MULTIMEDIA SYSTEM

張家瑋 博士

助理教授

國立臺中科技大學資訊工程系

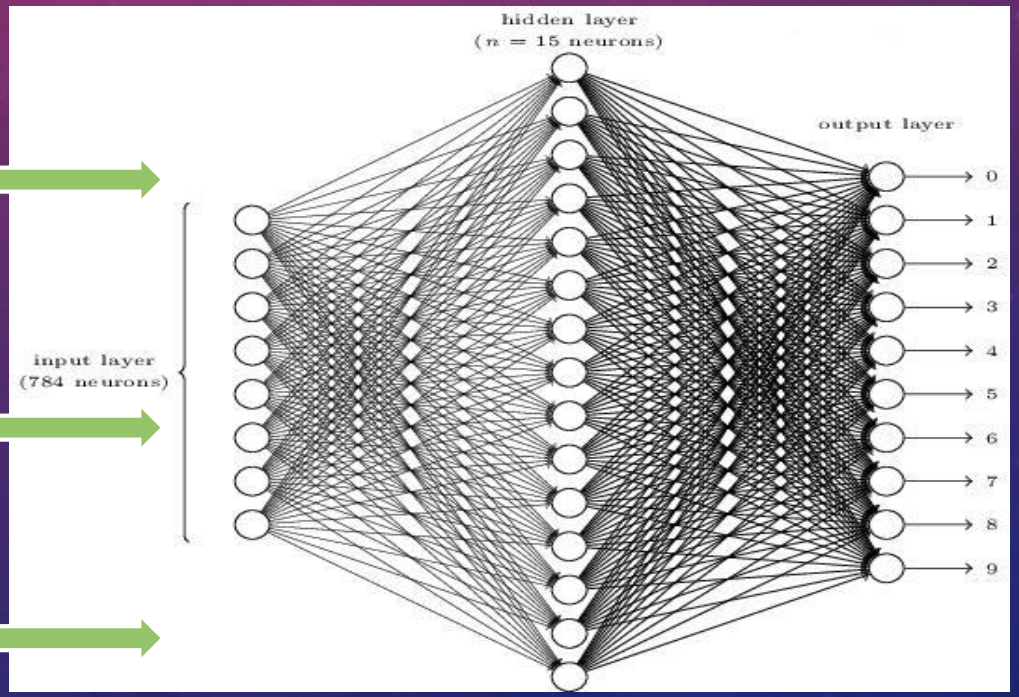
MNIST 手寫數字辨識



訓練資料

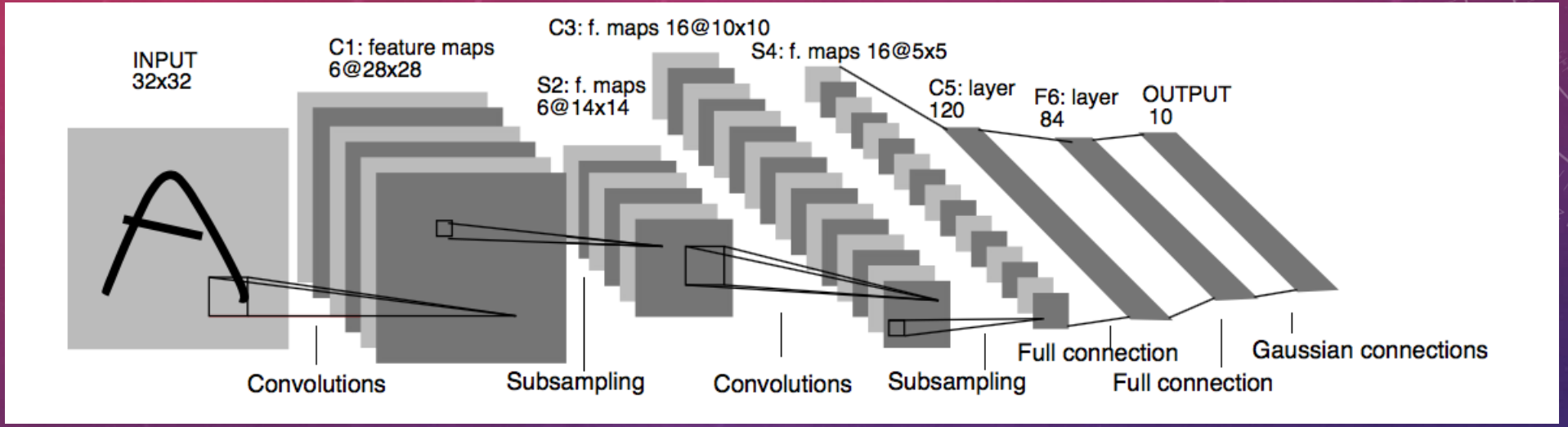
驗證資料

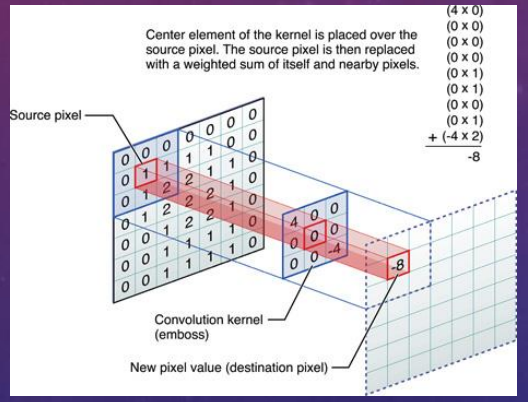
實際資料



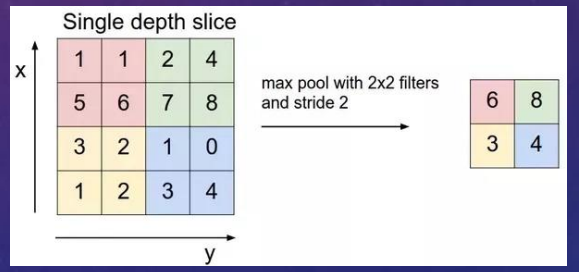
以最簡單的類神經網路架構，可達 91% 辨識率。若使用CNN則可高達 99% 辨識率。

卷積類神經網路





Convolution



Max Pooing

以 CNN 實作 - KERAS

- Step 1. 載入必要函式庫

```
import numpy as np
import matplotlib.pyplot as plt

from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import Conv2D, MaxPool2D, Flatten
from keras.utils import np_utils
```

- Step 2. 下載 MNIST 數據

```
nb_classes = 10
(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(type(x_train))
print("x_train shape", x_train.shape)
print("y_train shape", y_train.shape)
```

以 CNN 實作 - KERAS

- Step 3. 顯示圖片

```
fig = plt.figure()
plt.subplot(2,1,1)
plt.imshow(x_train[0], cmap="binary",
interpolation="none")
plt.title("image" + str(y_train[0]))
plt.subplot(2,1,2)
plt.hist(x_train[0].reshape(784))
plt.title("Pixel Values")
plt.show()
```

- Step 4. 準備訓練資料

```
img_size_x, img_size_y = 28, 28
x_train = x_train.reshape(x_train.shape[0], img_size_x, img_size_y, 1)
x_test = x_test.reshape(x_test.shape[0], img_size_x, img_size_y, 1)
input_shape = (img_size_x, img_size_y, 1)
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")
x_train /= 255
x_test /= 255
```

以 CNN 實作 - KERAS

- Step 5. 轉換為 One hot encoding

```
y_train = np_utils.to_categorical(y_train,nb_classes)
y_test = np_utils.to_categorical(y_test,nb_classes)
```

- Step 6. 定義類神經網路模型

Sequential可以讓我們按照順序將神經網路路串串起。深度學習為隱藏層有兩兩層或兩兩層以上.

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3,3), activation="relu", input_shape=input_shape))
model.add(Conv2D(64, kernel_size=(3,3), activation="relu"))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(10, activation="softmax"))
```

Loss:
https://keras.io/losses/
Optimizer:
https://keras.io/optimizers/

- Step 7. Compile

```
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

以 CNN 實作 - KERAS

- Step 8. 訓練模型

```
history = model.fit(x_train, y_train, batch_size=128, epochs=10, verbose=2,
validation_data=(x_test, y_test))
```

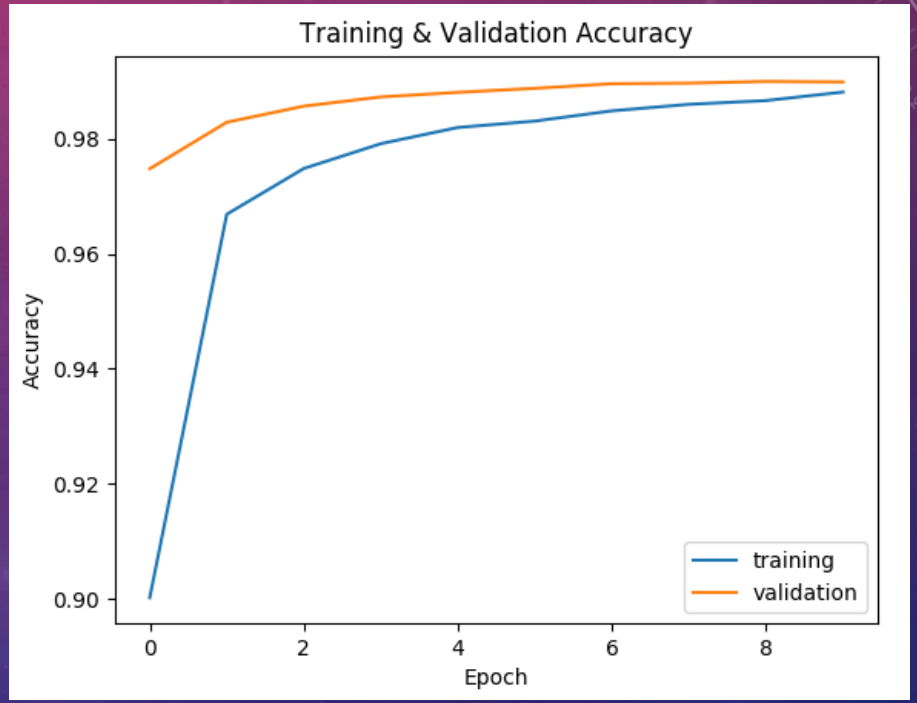
```
Epoch 1/10
125s - loss: 0.3322 - acc: 0.9002 - val_loss: 0.0789 - val_acc: 0.9748
Epoch 2/10
121s - loss: 0.1125 - acc: 0.9669 - val_loss: 0.0519 - val_acc: 0.9829
Epoch 3/10
123s - loss: 0.0844 - acc: 0.9748 - val_loss: 0.0424 - val_acc: 0.9857
Epoch 4/10
127s - loss: 0.0714 - acc: 0.9792 - val_loss: 0.0378 - val_acc: 0.9873
Epoch 5/10
124s - loss: 0.0617 - acc: 0.9820 - val_loss: 0.0364 - val_acc: 0.9881
Epoch 6/10
123s - loss: 0.0570 - acc: 0.9831 - val_loss: 0.0308 - val_acc: 0.9888
Epoch 7/10
124s - loss: 0.0506 - acc: 0.9849 - val_loss: 0.0294 - val_acc: 0.9896
Epoch 8/10
125s - loss: 0.0466 - acc: 0.9860 - val_loss: 0.0291 - val_acc: 0.9897
Epoch 9/10
124s - loss: 0.0441 - acc: 0.9867 - val_loss: 0.0286 - val_acc: 0.9900
Epoch 10/10
123s - loss: 0.0396 - acc: 0.9881 - val_loss: 0.0300 - val_acc: 0.9899
```

From 98% to 99%

以 CNN 實作 - KERAS

- Step 9. 檢查準確度

```
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training data')
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['training','validation'],loc='lower right')
plt.show()
```

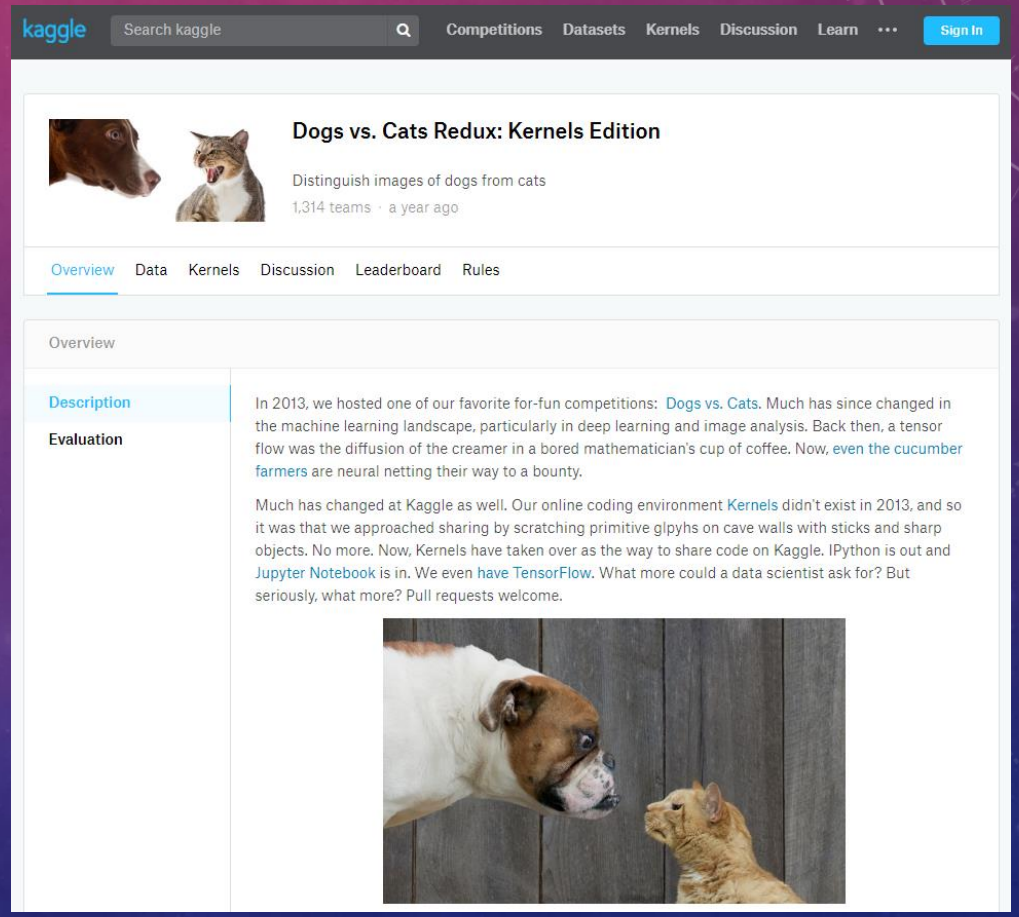


ADVANCED EXERCISE

DOG & CAT

DOGS VS. CATS

- https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition
  - Training data: 25000 images
  - Test data: 12500 images
  - For each image in the test set, you should predict a probability that the image is a dog (1 = dog, 0 = cat).
- 原始碼下載 [link]



以 CNN 實作 - TFLEARN

- Step 1. 載入必要函式庫

```
import cv2 # pip3 install opencv-python
import numpy as np
import os
from random import shuffle # 隨機資料庫
from tqdm import tqdm # 輸出進度條
import matplotlib.pyplot as plt # 繪圖
```

- Step 2. 設定檔案載入路徑與模型參數

```
train_dir = 'Pictures/train/'
test_dir = 'Pictures/test1/  '

img_size = 50
lr = 1e-3
```

以 CNN 實作 - TFLEARN

- Step 3. 取得圖片的label

```
def label_img(img):
    word_label = img.split('.')[-3]
    if word_label == 'cat': return [1,0]
    elif word_label == 'dog': return [0,1]
```

- Step 4. 創建訓練資料

```
def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(train_dir)):
        if (not img.endswith('.jpg')):
            continue
        label = label_img(img)
        path = os.path.join(train_dir, img)
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)  # 讀取圖片並轉為灰階
        img = cv2.resize(img, (img_size, img_size) )  # 將圖片轉為統一的大小
        training_data.append([np.array(img), np.array(label)])
    shuffle(training_data)
    return training_data

#---#

train_data = create_train_data()
```

以 CNN 實作 - TFLEARN

- Step 5. 創建測試資料

```
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(test_dir)):
        if (not img.endswith('.jpg')):
            continue
        path = os.path.join(test_dir,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (img_size, img_size))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    return testing_data
```

- Step 6. 匯入tflearn函式庫

```
import tflearn
# 需要安裝tensorflow，並安裝 tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
# 2維 CNN 以及最大池化
from tflearn.layers.core import input_data, dropout, fully_connected
# 輸入層，dropout，全連接層
from tflearn.layers.estimator import regression
# cross entropy層
```

以 CNN 實作 - TFLEARN

- Step 5. 創建測試資料

```
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(test_dir)):
        if (not img.endswith('.jpg')):
            continue
        path = os.path.join(test_dir,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (img_size, img_size))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    return testing_data
```

- Step 6. 匯入tflearn&tensorflow函式庫

```
import tflearn
# 需要安装tensorflow，並安裝 tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
# 2維 CNN 以及最大池化
from tflearn.layers.core import input_data, dropout, fully_connected
# 输入層，dropout，全連接層
from tflearn.layers.estimator import regression
# cross entropy層
import tensorflow as tf
# 引入 tensorflow
tf.reset_default_graph()
#初始化 tensorflow
```

以 CNN 實作 - TFLEARN

- Step 7. 建立CNN模型

```
convnet = input_data(shape = [None, img_size, img_size, 1], name = 'input')
convnet = conv_2d(convnet, 64, 5, activation='linear') # the number of convolutional filters, filter_size
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 32, 5, activation='linear')
convnet = max_pool_2d(convnet, 5)

convnet = conv_2d(convnet, 16, 5, activation='linear')
convnet = max_pool_2d(convnet, 5)

convnet = fully_connected(convnet, 16, activation = 'linear')
convnet = dropout(convnet, 0.5)

convnet = fully_connected(convnet, 8, activation = 'linear')

convnet = fully_connected(convnet, 4, activation = 'relu')

convnet = fully_connected(convnet, 2, activation='sigmoid')
convnet = regression(convnet, optimizer='adam', learning_rate = lr, loss='categorical_crossentropy', name='targets')
```

以 CNN 實作 - TFLEARN

- Step 8. 訓練CNN模型

```
model = tflearn.DNN(convnet, tensorboard_dir='log')

trainNum = -5000

train = train_data[:trainNum]
test = train_data[trainNum:]

X = np.array([i[0] for i in train], dtype=np.float64).reshape(-1, img_size, img_size, 1)
y = np.array([i[1] for i in train], dtype=np.float64)
Xtest = np.array([i[0] for i in test], dtype=np.float64).reshape(-1, img_size, img_size, 1)
ytest = np.array([i[1] for i in test], dtype=np.float64)

model.fit({'input': X}, {'targets': y}, n_epoch=10, batch_size=250, validation_set=({'input': Xtest}, {'targets': ytest}),
snapshot_step=500, show_metric=True, run_id='model' )
```

以 CNN 實作 - TFLEARN

- Step 9. 測試CNN模型

```
test_data = process_test_data()

fig = plt.figure()
for num,data in enumerate(test_data[:16]):
    img_num = data[1]
    img_data = data[0]
    y = fig.add_subplot(4, 4, num+1)
    orig = img_data
    data = img_data.reshape(img_size, img_size, 1)
    model_out = model.predict([data])[0]
    print(model_out)
    if np.argmax(model_out) == 1:
        label = 'Dog'
    else:
        label = 'Cat'

    y.imshow(orig, cmap='gray')
    plt.title(label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)

plt.tight_layout()
plt.show()
```

```
[0.14723705 0.98256   ]
[0.04469623 0.99915826]
[0.99998796 0.00834211]
[0.99993527 0.01661933]
[0.99877125 0.05415697]
[0.43036035 0.6431105 ]
[0.20522958 0.95668554]
[0.8880429  0.28058085]
[0.08643436 0.9956601 ]
[0.99908304 0.04826429]
[0.9999168  0.01841162]
[0.1397599  0.98483086]
[0.99558544 0.08876048]
[0.43750185 0.6269166 ]
[0.10178897 0.9934009 ]
[0.10858331 0.9922093 ]
```

THANK YOU