Keras
Introduction & Installation

Guan-Shiuan Kuo

Introduction

Deep Learning framework

What is Keras ?

- Keras is a Python wrapper around Theano and TensorFlow.
- Hide many low-level operations that you may not care about.
- Sacrifice some functionality of Theano and TensorFlow for much easier user interface.

User Experience

http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017_2/Lecture/keras.pdf

Installation

Installation via terminal

0. 安裝

在這邊推薦大家使用OSX或是Linux來運行keras，雖然在windows的環境下依然可以運行keras，但在安裝上需要依賴過多，與對gpu、記憶體控管等等的支持較差，易照成不可預期的錯誤。

在這我們使用python3與Tensorflow backend進行教學

i. Install on OSX/Linux

在安裝keras前，請先安裝Tensorflow
安裝的方法請見Tensorflow基礎篇

在你的終端機(terminal)輸入

```
###安裝keras
sudo pip3 install keras
###安裝存儲model的套件
sudo pip3 install h5py
```

###安裝keras
sudo pip3 install keras
###安裝存儲model的套件
sudo pip3 install h5py

ii. Install on Windows

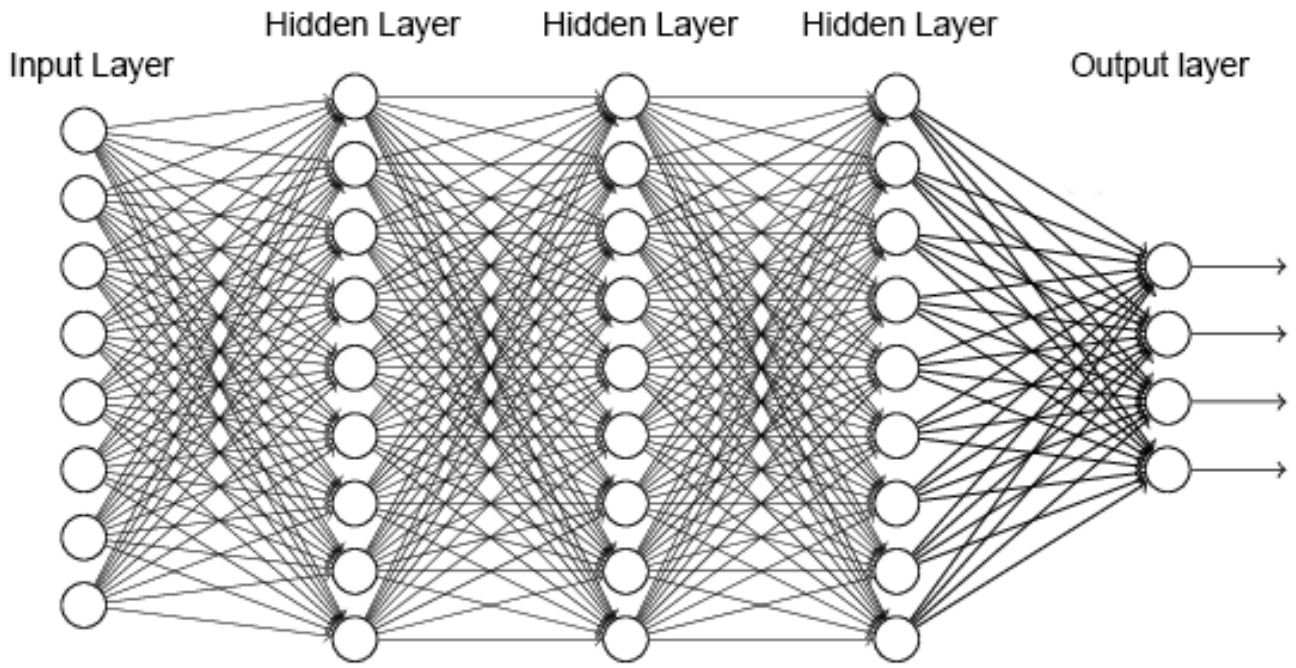
筆者再一次建議大家盡量別在Windows下運行keras，若是不熟悉Linux的使用，大家可以在Windows下寫code，再將code傳到Linux上去運行。

詳細安裝方法：Windows環境下使用Anaconda執行Python環境

https://fgc.stpi.narl.org.tw/activity/videoDetail/4b1141305d9cd231015d9d03cfd10027

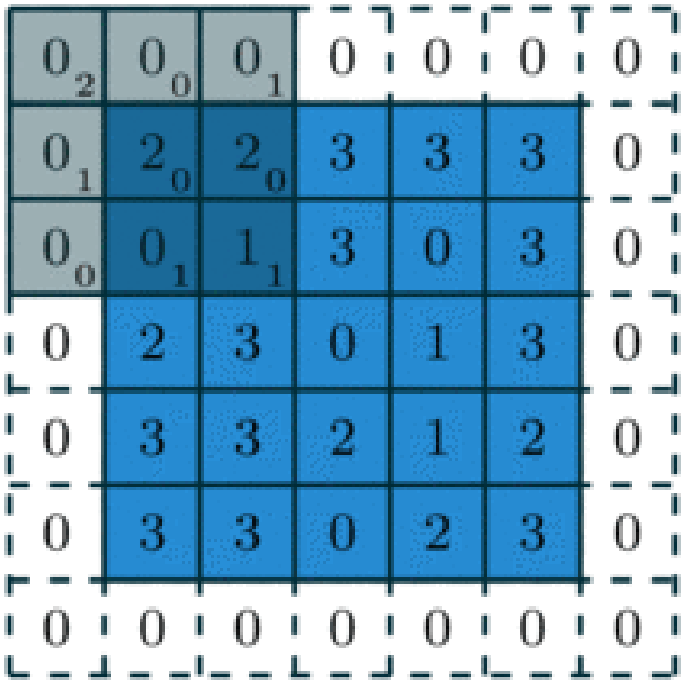
Review

Dense (Fully-Connected) Layer



https://medium.com/the-almanac/an-executive-guide-to-artificial-intelligence-part-3-b550ee1578b8

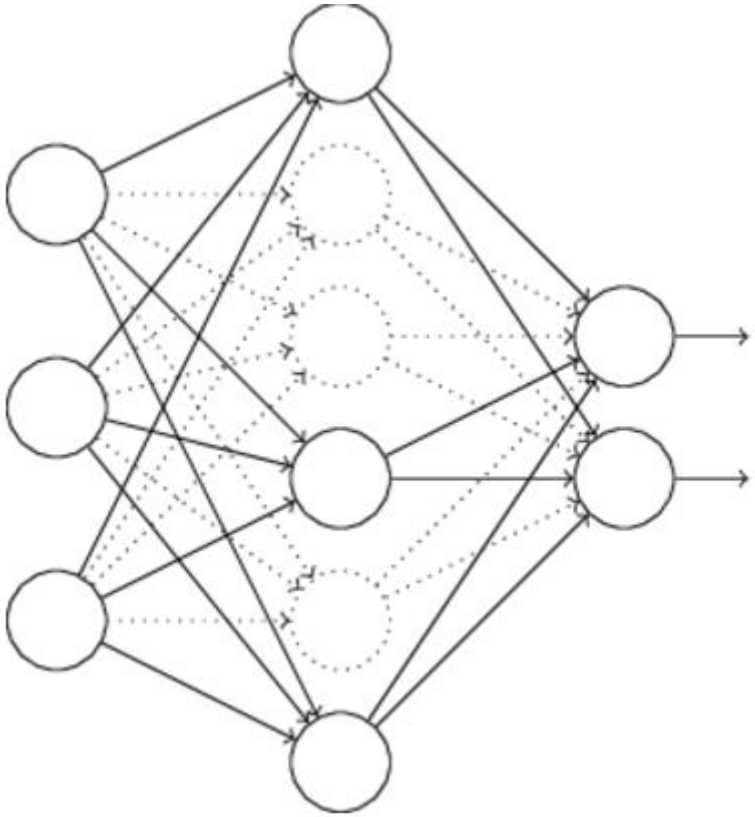
Convolutional Layer



1	6	5
7	10	9
7	10	8

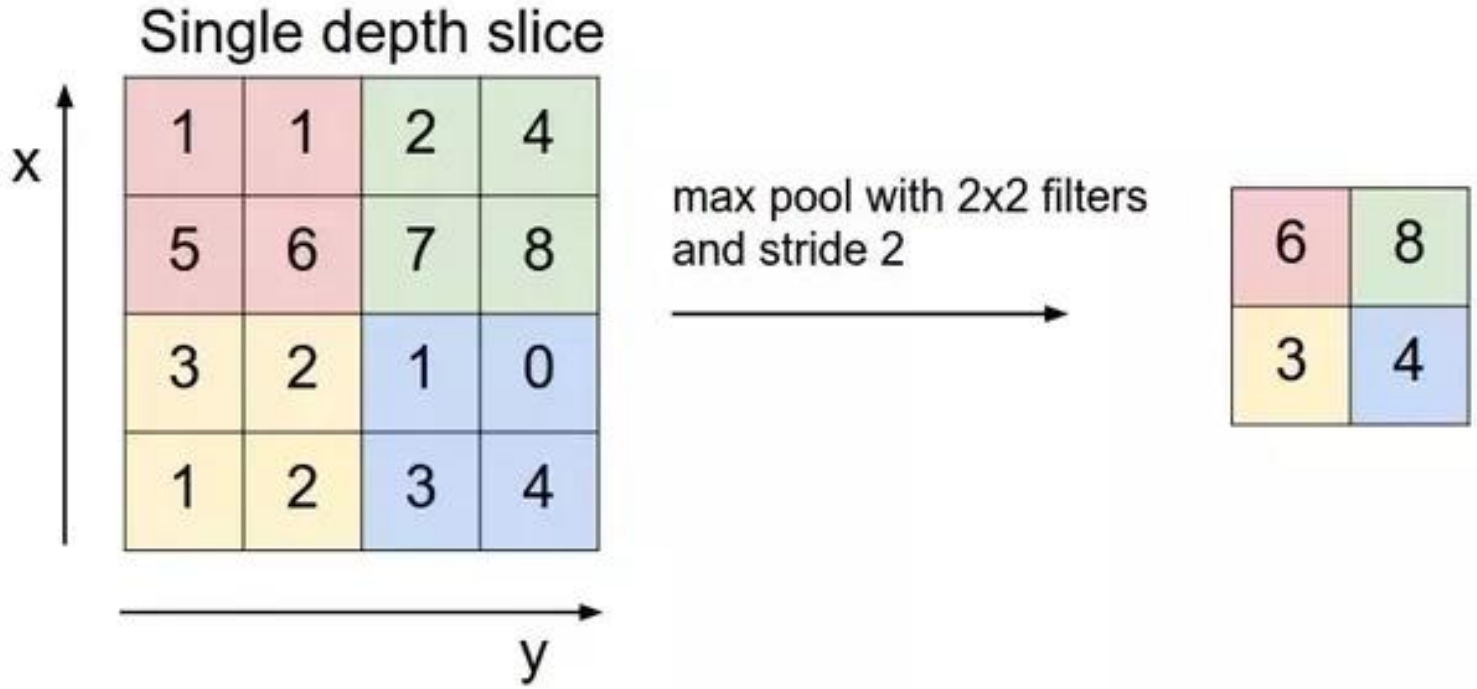
http://deeplearning.net/software/theano_versions/dev/tutorial/conv_arithmetic.html

Dropout Layer



https://www.quora.com/In-TensorFlow-what-is-a-dense-and-a-dropout-layer

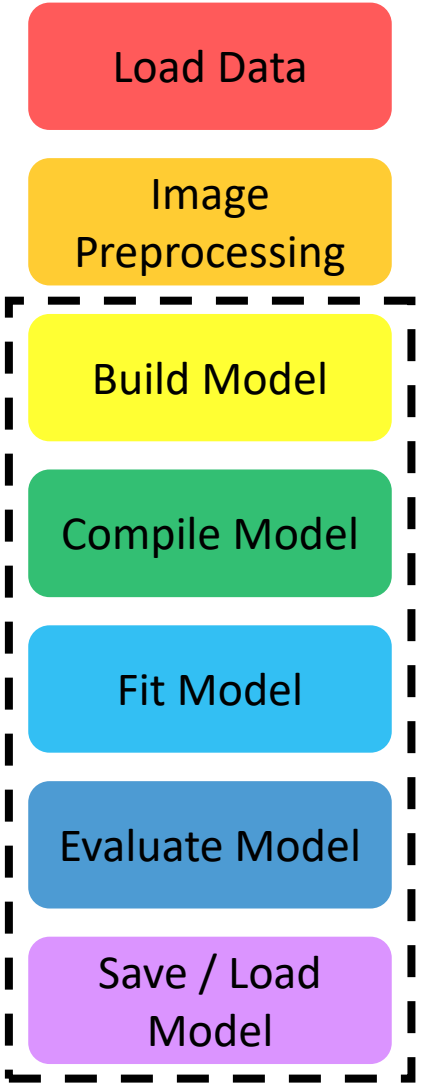
Pooling Layer



https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks

Training Procedure

Flow Chart


Load Data
Image Preprocessing
Build Model
Compile Model
Fit Model
Evaluate Model
Save / Load Model

Build Model

```
def myCNN():
    img_input = Input(shape=(28, 28, 1))
    co1 = Conv2D(10, (5, 5), padding='valid', activation='relu', name='co1')(img_input)
    do1 = Dropout(0.2, name='do1')(co1)
    co2 = Conv2D(10, (5, 5), padding='valid', activation='relu', name='co2')(do1)
    do2 = Dropout(0.2, name='do2')(co2)
    flat = Flatten()(do2)
    fc1 = Dense(num_classes, activation='softmax', name='do3')(flat)

    model = Model(img_input, fc1)
    L_model = Model(img_input, co1)
    H_model = Model(img_input, do2)

    return model, L_model, H_model
```

Compile Model

- Choose loss function, optimizer, and evaluating metrics.

```
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=1e-4),
              metrics=['accuracy'])
```

Fit & Save Model

- Train the model using the optimizer.
- Specify training epochs and batch size.

```
epochs = 20
batch_size = 256

history = model.fit(x_train,
                    y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    validation_data=(x_valid, y_valid),
                    callbacks=cb,
                    verbose=1)
```

```
ckpt = ModelCheckpoint('CNN_model_e{epoch:02d}',
                       monitor='val_acc',
                       save_best_only=False,
                       save_weights_only=True,
                       verbose=1)
cb = [ckpt]
```

Evaluate Model

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/20
50000/50000 [==============================] - 41s 815us/step - loss: 1.1859 - acc: 0.6690 - val_loss: 0.4
154 - val_acc: 0.8830

Epoch 00001: saving model to CNN_model_e01
Epoch 2/20
50000/50000 [==============================] - 41s 820us/step - loss: 0.4303 - acc: 0.8714 - val_loss: 0.3
010 - val_acc: 0.9154

Epoch 00002: saving model to CNN_model_e02
Epoch 3/20
50000/50000 [==============================] - 41s 828us/step - loss: 0.3396 - acc: 0.8987 - val_loss: 0.2
507 - val_acc: 0.9307

Epoch 00003: saving model to CNN_model_e03
Epoch 4/20
50000/50000 [==============================] - 43s 864us/step - loss: 0.2842 - acc: 0.9158 - val_loss: 0.2
129 - val_acc: 0.9407

Epoch 00004: saving model to CNN_model_e04
Epoch 5/20
50000/50000 [==============================] - 45s 907us/step - loss: 0.2436 - acc: 0.9288 - val_loss: 0.1
854 - val_acc: 0.9487

Epoch 00005: saving model to CNN_model_e05
Epoch 6/20
50000/50000 [==============================] - 43s 862us/step - loss: 0.2108 - acc: 0.9384 - val_loss: 0.1
644 - val_acc: 0.9546
```

Load & Evaluate Model

```
model_name = 'CNN_model_e20'
model.load_weights(model_name)

# prediction = model.predict(x_test)
accuracy = model.evaluate(x_test, y_test, verbose=0)
print('Testing accuracy: {}'.format(accuracy))
```

```
Testing accuracy: [0.0704437150507234, 0.9795]
```

Tips

Load Data

- Normalize
  - Divide pixel value by 255
  - Minus mean and divide by std

Model

- Use dropout layer between dense layer.
- Use “relu” or “tanh” as the activation function
- Use “softmax” as the activation of the last dense layer