JERRY WU & NTPU TEAM

VQA實作

```
%matplotlib inline
import os, argparse
import cv2, spacy, numpy as np
from keras.models import model_from_json
from keras.optimizers import SGD
from sklearn.externals import joblib
```

- 將所需的資料庫import進去

```
VQA_model_file_name      = 'models/VQA/VQA_MODEL.json'
VQA_weights_file_name    = 'models/VQA/VQA_MODEL_WEIGHTS.hdf5'
label_encoder_file_name  = 'models/VQA/FULL_labelencoder_trainval.pkl'
CNN_weights_file_name    = 'models/CNN/vgg16_weights.h5'
```

- 下載需要的CNN模型

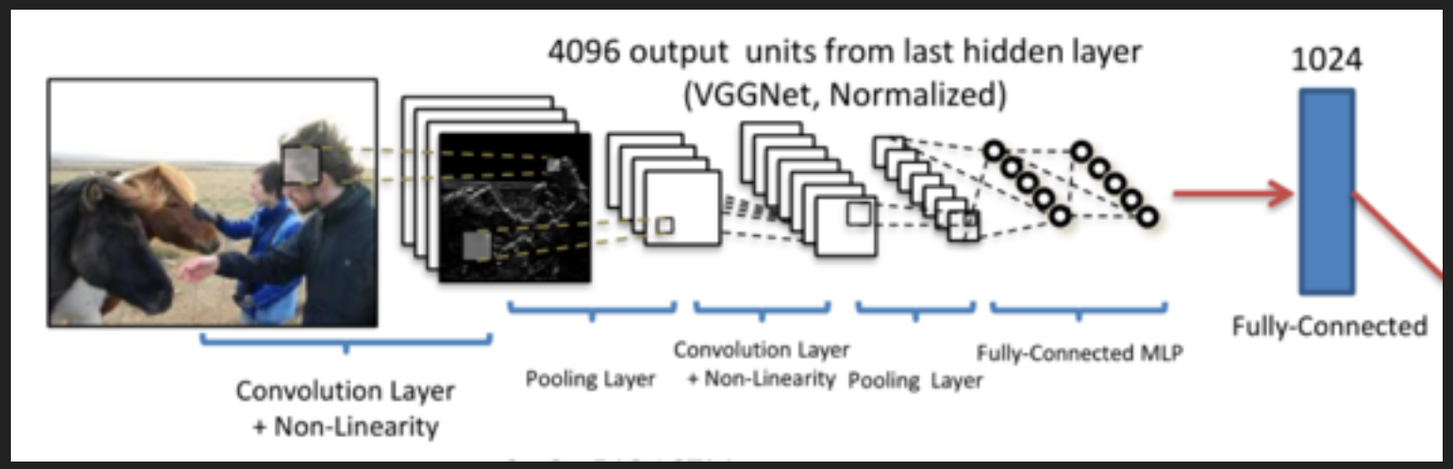
文字

```python
def get_image_model(CNN_weights_file_name):
    ''' Takes the CNN weights file, and returns the VGG model update
    with the weights. Requires the file VGG.py inside models/CNN '''
    from models.CNN.VGG import VGG_16
    image_model = VGG_16(CNN_weights_file_name)

    # this is standard VGG 16 without the last two layers
    sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
    # one may experiment with "adam" optimizer, but the loss function for
    # this kind of task is pretty standard
    image_model.compile(optimizer=sgd, loss='categorical_crossentropy')
    return image_model
```

- 定義CNN模型
- 使用基本計算loss的函數

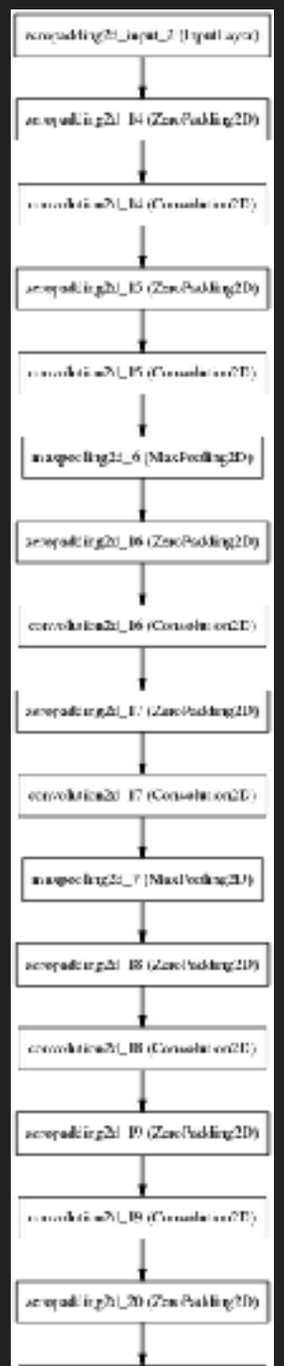
文字


4096 output units from last hidden layer
(VGGNet, Normalized)
1024
Convolution Layer
+ Non-Linearity
Pooling Layer
Convolution Layer
+ Non-Linearity
Pooling Layer
Fully-Connected MLP
Fully-Connected

- VGG 基本架構圖

文字

```
In [13]: from keras.utils.visualize_util import plot
         model_vgg = get_image_model(CNN_weights_file_name)
         plot(model_vgg, to_file='model_vgg.png')
```



文字

```python
def get_image_features(image_file_name, CNN_weights_file_name):
    ''' Runs the given image_file to VGG 16 model and returns the
    weights (filters) as a 1, 4096 dimension vector '''
    image_features = np.zeros((1, 4096))
    # Magic_Number = 4096  > Comes from last layer of VGG Model

    # Since VGG was trained as a image of 224x224, every new image
    # is required to go through the same transformation
    im = cv2.resize(cv2.imread(image_file_name), (224, 224))
    im = im.transpose((2,0,1)) # convert the image to RGBA


    # this axis dimension is required because VGG was trained on a dimension
    # of 1, 3, 224, 224 (first axis is for the batch size
    # even though we are using only one image, we have to keep the dimensions consistent
    im = np.expand_dims(im, axis=0)

    image_features[0,:] = get_image_model(CNN_weights_file_name).predict(im)[0]
    return image_features
```

- 定義image feature 的大小
- resize(reshape)

文字

```
def get_question_features(question):
    ''' For a given question, a unicode string, returns the time series vector
    with each word (token) transformed into a 300 dimension representation
    calculated using Glove Vector '''
    word_embeddings = spacy.load('en', vectors='en_glove_cc_300_1m_vectors')
    tokens = word_embeddings(question)
    question_tensor = np.zeros((1, len(tokens), 300))
    for j in xrange(len(tokens)):
            question_tensor[0,j,:] = tokens[j].vector
    return question_tensor
```

- 自然語言處理
- Word2Vec

文字

```
In [16]: word_embeddings = spacy.load('en', vectors='en_glove_cc_300_1m_vectors')

In [17]: obama = word_embeddings(u"obama")
         putin = word_embeddings(u"putin")
         banana = word_embeddings(u"banana")
         monkey = word_embeddings(u"monkey")
```

- Word2Vec範例

```
In [18]: obama.similarity(putin)
Out[18]: 0.435141125341493 85
```

```
In [19]: obama.similarity(banana)
Out[19]: 0.17831375020636123
```

```
banana.similarity(monkey)
0.45207779162154438
```

文字

```
In [21]: def get_VQA_model(VQA_model_file_name, VQA_weights_file_name):
             ''' Given the VQA model and its weights, compiles and returns the model '

             # thanks the keras function for loading a model from JSON, this becomes
             # very easy to understand and work. Alternative would be to load model
             # from binary like cPickle but then model would be obfuscated to users
             vqa_model = model_from_json(open(VQA_model_file_name).read())
             vqa_model.load_weights(VQA_weights_file_name)
             vqa_model.compile(loss='categorical_crossentropy', optimizer='rmsprop')
             return vqa_model
```

- 圖像處理的CNN模型+語言處理模型放入VQA模型中

實際測試

```
In [ ]: image_file_name = 'test.jpg'
        question = u"What vehicle is in the picture?"
```

```
In [ ]: # get the image features
        image_features = get_image_features(image_file_name, CNN_weights_file_name)

In [ ]: # get the question features
        question_features = get_question_features(question)
```

- 取得圖片資訊
- 取得問題資訊

文字

```
y_output = model_vqa.predict([question_features, image_features])

# This task here is represented as a classification into a 1000 top answers
# this means some of the answers were not part of training and thus would
# not show up in the result.
# These 1000 answers are stored in the sklearn Encoder class
labelencoder = joblib.load(label_encoder_file_name)
for label in reversed(np.argsort(y_output)[0,-5:]):
    print str(round(y_output[0,label]*100,2)).zfill(5), "% ", labelencoder.inverse_transform(label
)
```

- 輸出結果

78.32 % train
01.11 % truck
00.98 % passenger
00.95 % fire truck
00.68 % bus

- 使用URL做圖片分析
- 就要新增 `from skimage import io`

文字

```
In [ ]: image_file_name = "http://www.newarkhistory.com/indparksoccerkids.jpg"
        # get the image features
        image_features = get_image_features(image_file_name, CNN_weights_file_name)
```

- 丟一個問題

```
In [ ]: question = u"What are they playing?"

        # get the question features
        question_features = get_question_features(question)
```

- 輸出結果

40.52 % tennis
28.45 % soccer
17.88 % baseball
11.67 % frisbee
00.15 % football

- 換個問題

```
In [ ]: question = u"Are they playing soccer?"

        # get the question features
        question_features = get_question_features(question)
```

- 輸出結果

93.15 % yes
06.42 % no
00.02 % right
00.01 % left
000.0 % man