本次用 tensorflow 做基礎辨識，我們學到了以下幾點知識：

1. TensorFlow 是由 Google 提供的開放原始碼程式庫。Google 的很多產品，早就使用 TensorFlow 技術開發深度學習與機器學習功能，例如:Gmail 過濾垃圾信、Google 語音辨識、Google 圖片辨識、Google 翻譯..等。

2. TensorFlow 是一個機器學習的開發平台，可提供使用者以 ANN, CNN, SVM 等機器學習演算法開發程式。可使用 Python 與 C++運行 Tensorflow，但較多開發者使用 Python 另外 TensorFlow 有 cpu 與 gpu 兩個版本，訓練大型的網路時，一般會選用 gpu 的版本訓練，速度約快數十倍。

3. 人工神經網路（英語：Artificial Neural Network，ANN），簡稱神經網路（Neural Network，NN）或類神經網路，在機器學習和認知科學領域，是一種模仿生物神經網路（動物的中樞神經系統，特別是大腦）的結構和功能的數學模型或計算模型，用於對函式進行估計或近似。神經網路由大量的人工神經元聯結進行計算。

簡單來說，我們了解到 tensorflow 的運用以及其主要運用的領域維類神經網路，並且多開發者使用 Python 來處理之。

Tensorflow 的運用？

1. TensorFlow 的進化，電腦現在能夠以更接近人類的方法學習
「深度學習」改變了機器學習的方法。我們再也不需要定義任何的「特點」，我們要做的是提供大量的資訊給機器，以上面的例子來說，就是輸入大量的「狗」和「貓」的圖片。然後告訴機器，哪一張圖片是「狗」，哪一張是「貓」。而它就能夠自行定義出牠們不一樣的地方。當機器成功建立了這個「模型」後，它就能夠分辨任何一張圖片上的「狗」與「貓」。

2. 不用整理，電腦也能夠為你找出「衝浪」與「滑雪」的照片

Google 相簿就使用 TensorFlow 讓我們快速的使用關鍵字查詢未分類或沒有 Tag 的照片。因為 TensorFlow 已在網上透過分類與 Tag 學習到了各種關鍵字。因此，當你輸入「海」、「雪地」或「黃昏」，TensorFlow 就會找出對應的照片。

3. 只需要 3 天的訓練，電腦就可以幫你分類小黃瓜

系統為傳送帶上的小黃瓜拍照，然後 TensorFlow 根據上傳的照片判斷小黃瓜的類別。判定後，通知「機器手臂」將小黃瓜推進標誌了不同類別的箱子裡。

本次實作 tensorflow 遇到的困難：

Rouyun Pan Follow
Aug 6, 2017 · 3 min read

用Tensorflow實現物件辨識

花了兩三天用Tensorflow 兜了一個經典的Alexnet 模型, 來做物件辨識, 原本Alexnet模型的作者 kratzert 用 Imagenet 資料集, 訓練了一千個物件的分類, 這篇我利用遷移學習, 不用重新訓練整個權重, 利用它預訓練的權重, 只重新定義後面兩層的分類器, 以新的資料集來訓練新的分類器權重, 做了一個貓狗分類器

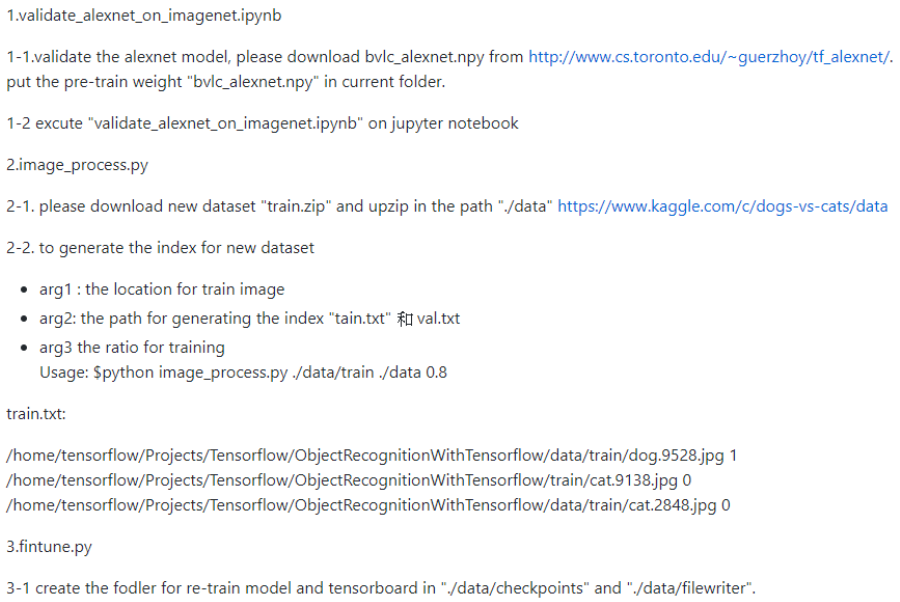
*Source code:
https://github.com/rouyunpan/ObjectRecognitionWithTensorflow

*環境配置:
- Tensorflow 1.2
- Python 3.5
- Alexnet 模型 (以Imgnet 資料集預訓練的權重 bvlc_alexnet.npy)
http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/
- kaggle 的貓狗照片資料集
https://www.kaggle.com/c/dogs-vs-cats

*Alexnet 是2012 年深度學習的開山大作, 是當年Imagenet 大賽的冠軍. 後面幾年冠軍都是演化這樣的CNN模型了.

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

本來運用其他人的在 github 上的資料，想說來執行一次 tensorflow，觀察其辨識程度，結果在設定路徑上發現一些困難，以致於資料無法讀取



1.validate_alexnet_on_imagenet.ipynb

1-1.validate the alexnet model, please download bvlc_alexnet.npy from http://www.cs.toronto.edu/~guerzhoy/tf_alexnet/.
put the pre-train weight "bvlc_alexnet.npy" in current folder.

1-2 excute "validate_alexnet_on_imagenet.ipynb" on jupyter notebook

2.image_process.py

2-1. please download new dataset "train.zip" and upzip in the path "./data" https://www.kaggle.com/c/dogs-vs-cats/data

2-2. to generate the index for new dataset

- arg1 : the location for train image
- arg2: the path for generating the index "tain.txt" 和 val.txt
- arg3 the ratio for training
  Usage: $python image_process.py ./data/train ./data 0.8

train.txt:

/home/tensorflow/Projects/Tensorflow/ObjectRecognitionWithTensorflow/data/train/dog.9528.jpg 1
/home/tensorflow/Projects/Tensorflow/ObjectRecognitionWithTensorflow/train/cat.9138.jpg 0
/home/tensorflow/Projects/Tensorflow/ObjectRecognitionWithTensorflow/data/train/cat.2848.jpg 0

3.fintune.py

3-1 create the fodler for re-train model and tensorboard in "./data/checkpoints" and "./data/filewriter".

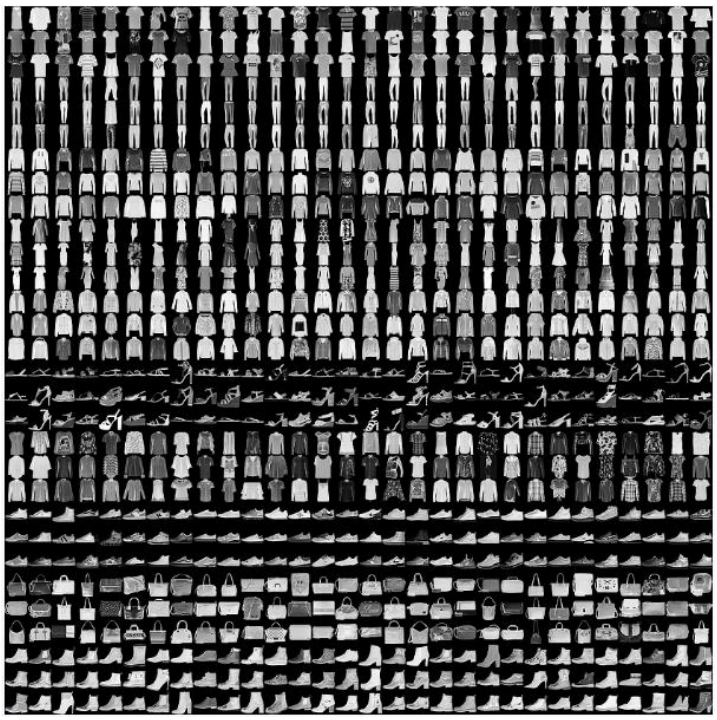
主要是在第 3 點發現困難，我們還下載了很多 python 的其他套件以及 jupyter notebook 以執行 validate_alexnet_on_imagenet.ipynb 但圖片就是無法讀取，故我們最後在官網看了一套教學，才達成這次 tensorflow 的執行。

本次的機器學習實作是使用人工神經網路來分辨各種服飾的圖片。

1. Import keras. Keras 是一個 Tensorflow 裡可以用來建立訓練模型的 API，也就是幫助使用者建立完善的神經網路，以輕鬆的開發深度學習。

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

2. Import the Fashion MNIST dataset. MNIST 是一個入門級的計算機視覺數據集，它包含各種手寫數字圖片:



這個圖片庫裡總共有 60,000 張圖片、10 種不同服飾，分別以 0~9 編號，每張圖片的像素為 28*28。

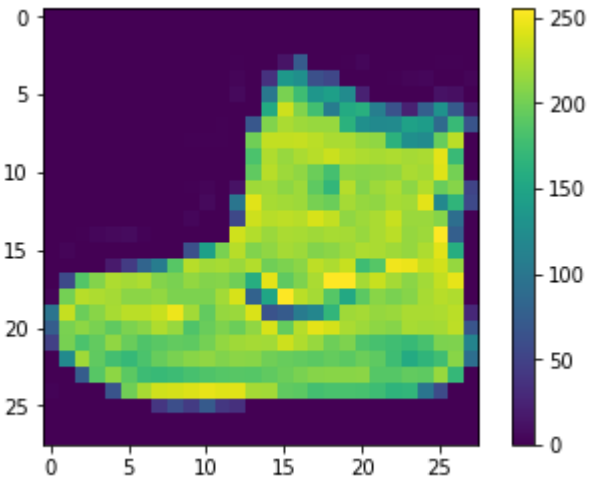
```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

3. 印出模型資訊。本次機器學習實作總共有 60000 張訓練圖片，我們將使用這 60000 張圖片來當作訓練材料，而測試資料有 10000 張圖片，這些則是用來驗收訓練的成果。

```
train_images =  (60000, 28, 28)
train_label size =  60000
train_label =  [9 0 0 ... 3 0 5]
test_images =  (10000, 28, 28)
test_label size =  10000
test_label =  [9 2 1 ... 8 1 5]
```

4. Preprocess the data. 我們現在要正式開始機器學習，在開始之前，所有的圖片都要經過預處理的程序，使得資料可以標準化:

```
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
```



這些圖片的像素值落於 0~255 之間。在將圖片放入神經網路模型之前，我們將像素值縮至 0~1 之間，所以將所有像素值除以 255:

```
train_images = train_images / 255.0

test_images = test_images / 255.0
```

接著顯示前 25 張圖片的屬性，確認每張圖片都符合格式就可以開始建神經網路了:

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
```

5. Build the model. 神經網路是由很多的 layer(層)組成，而每一層都有許多的神經元，然後層與層之間會連接在一起，形成龐大的網路。

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

接著我們增加一些 tool:

Loss function 可以幫忙計算這次訓練的精準度大概的百分比。metrics 可以顯示訓練的進度條和每次訓練完的 loss function。這些功能需要引用 optimizer:

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

6. Train the model. 將訓練資料送入神經網路模型，寫入這一行，可以呼叫訓練指令:

```
model.fit(train_images, train_labels, epochs=5)
```

訓練完後，訓練進度、失準度、精準度將會顯示在螢幕上:

```
Epoch 1/5
60000/60000 [==============================] - 5s 85us/step - loss: 0.4948 - acc: 0.8269
Epoch 2/5
60000/60000 [==============================] - 5s 84us/step - loss: 0.3731 - acc: 0.8659
Epoch 3/5
60000/60000 [==============================] - 5s 83us/step - loss: 0.3350 - acc: 0.8777
Epoch 4/5
60000/60000 [==============================] - 5s 84us/step - loss: 0.3104 - acc: 0.8850
Epoch 5/5
60000/60000 [==============================] - 5s 87us/step - loss: 0.2921 - acc: 0.8935
```

7. Evaluate accuracy. 接著我們將測試圖片丟入模型，看看模型是否訓練成功:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)

print('Test accuracy:', test_acc)
```

```
10000/10000 [==============================] - 1s 50us/step
Test accuracy: 0.8712
```

我們可以從精準度看出，原本將訓練圖片丟入模型的精準度最後大概有到 89%，但是將測試圖片丟入模型後，精準度卻降到 87%，這種情況稱為過適 (overfitting)，原因有可能是訓練資料太少或是模型太複雜。

8. Make predictions. 模型訓練完後，我們要來做預測，確定訓練的成果:

```
predictions = model.predict(test_images)
```

```
i =  0
Prediction array =  [5.2920160e-07 1.6641306e-07 1.6110489e-07 3.8030510e-07 2.3757478e-07
 1.0831458e-02 1.4625444e-06 1.6053805e-02 6.8209411e-06 9.7310501e-01]
Prediction =  9
test_image label =  9
i =  1
Prediction array =  [1.0472421e-05 7.5954833e-11 9.9814212e-01 1.2868036e-07 1.2855196e-03
 2.6816183e-10 5.6179793e-04 3.6860718e-12 4.8338229e-08 6.2671521e-14]
Prediction =  2
test_image label =  2
i =  2
Prediction array =  [3.8562607e-07 9.9999774e-01 1.6291849e-09 1.8767988e-06 3.5280301e-08
 7.4819382e-13 3.8012424e-09 1.9838871e-19 4.6019733e-11 2.1493243e-15]
Prediction =  1
test_image label =  1
i =  3
Prediction array =  [1.0521499e-06 9.9986279e-01 3.6366576e-08 1.3562270e-04 3.6211381e-07
 3.7073462e-11 7.3905071e-08 2.4475414e-16 6.5401928e-10 2.9238703e-12]
Prediction =  1
test_image label =  1
i =  4
Prediction array =  [3.0497271e-01 8.2382794e-06 4.1829977e-02 6.3047912e-03 1.0625788e-02
 1.6475262e-05 6.3516200e-01 1.6013810e-06 1.0777676e-03 7.8709866e-07]
Prediction =  6
test_image label =  6
i =  5
Prediction array =  [8.1834578e-06 9.9997938e-01 3.7459438e-07 8.4110497e-06 2.2372715e-06
 5.0896700e-11 1.3371952e-06 8.8328748e-15 1.8127592e-09 3.5871809e-12]
Prediction =  1
test_image label =  1
i =  6
Prediction array =  [1.2497189e-04 6.0445604e-05 2.2552580e-02 3.5147727e-03 9.4775778e-01
 4.1068166e-08 2.5958244e-02 1.2415293e-08 3.1053125e-05 1.3016557e-07]
Prediction =  4
test_image label =  4
i =  7
Prediction array =  [3.5810564e-05 1.9597049e-08 2.1422072e-03 1.2420914e-04 2.6977204e-02
 1.9831608e-08 9.7071683e-01 1.3022213e-09 3.6759509e-06 9.2126040e-09]
Prediction =  6
test_image label =  6
i =  8
Prediction array =  [5.2314459e-05 1.8135680e-06 1.4200521e-05 6.0307644e-07 4.3291757e-06
 9.9988222e-01 4.6145979e-06 3.6341338e-05 2.9376147e-06 5.0650539e-07]
Prediction =  5
test_image label =  5
```

如果訓練成功的話，可以看到所有的 prediction 都會和 test_image　label 吻合，表示預測成功，我們完成了一次成功的訓練!

心得: 當初因為我們兩個都對機器學習有興趣，所以上網查了一下資料後找到了最近 google 很關注的 tensorflow，所以決定來玩玩看這個東西。沒想到的是，別說寫出程式碼，我們從一開始的安裝就陷入瓶頸，我看網路上也有很多人遇到安裝的問題，因為安裝 tensorflow 有滿繁雜的流程，需要安裝很多其他工具，還會牽扯到版本不符以及檔案位置的問題，所以真的花了一段時間才安裝完成。雖然程式碼是從 tenrsorflow 官網複製來的，並沒有自己寫，不過我們至少大致了解為什麼要這麼寫，以及其中的道理，也順利跑出了正確的結果。如果以後有機會的話，希望可以更精通 tensorflow，玩出自己想做的機器學習!