

手機刮痕辨識(Final Project)

F74062044 黃政瑋

F74066022 徐澤淞

F74064054 戴宏諺

背景與動機(Abstract)

這堂課教了我們人工智慧，從一開始的 Gradient Descent 的連鎖率數學公式推導(了解了更新參數的機制)，到後來的 Learning Rate 影響收斂速度，慢慢到 CNN 的 Filter、Convolution、Deconvolution，甚至是 RNN，Reinforcement learning 等等，學了很多的機器學習概念。

我們也看了以前完全沒有接觸的論文，原本想說論文離我們太遙遠，應該不需要這麼早看論文，畢竟我們才大二，但在實作 Final Project 後，發現論文看到的東西都是當下的養分，我們可以很清楚建立模型的流程，Batch Normalization 的重要性，甚至是 Deconvolution 的應用等等，獲益匪淺，沒有白花看論文的時間。

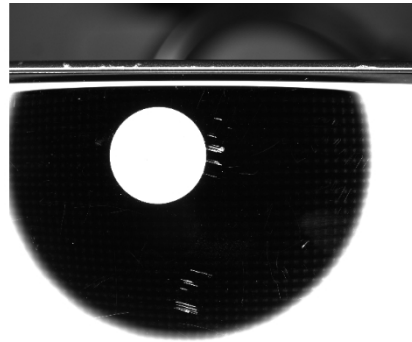
我們想要利用 Final Project 來實踐上面所學的概念，這次老師給我們的是手機刮痕辨識的專案，跟以往我們看論文的一些主題不太一樣，這次是對於手機照片進行切割，是一個很有挑戰的題目，也讓我們充滿鬥志。

問題描述(Introduction)

這次我們要做的是手機刮痕辨識，輸入一張手機的照片，我們利用深度學習訓練 Model 來準確的預測刮痕，將刮痕的地方辨識出來，輸出一張背景為黑色並將預

測的刮痕標示為白色。

(下圖為手機原圖)



(下圖為預計產生的 ground truth)



潛在應用與價值

應用一

辨識瑕疵亦或是刮痕，對於工廠來說想必是很重要的，如果有新的產品需要銷售，必須要檢測瑕疵，如果雇用很多的人力去看，很花金錢和時間，如果我們可以用機器學習進行辨識，可以省掉很多人力支出，並且透過機器辨識可以加速檢驗商

品的速度，如果我們可以將這個技術做好，將具有很大的商業價值。

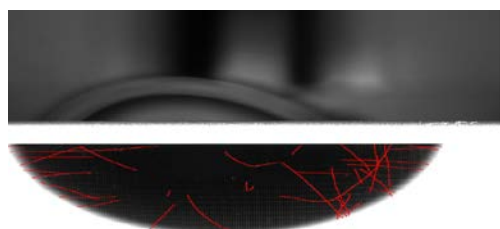
應用二

當手機公司想要回收舊機來折現金，但畢竟人工辨識有失公平性、準確度，這時候可以利用我們做出的這項技術，來辨識手機的回損率，針對回損率來決定舊機折現的金額。

資料描述

這次助教有先將資料分配給每一組，平均每一組標記兩到三個資料夾，針對手機原圖(大約是 2000*2000pixel 數相當高)利用繪圖軟體，將人眼便認為刮痕的地方利用紅色標記出來。

標記出來後，我們可以利用 python 這個程式語言，將人工標記的照片中紅色的轉為白色，其餘背景為黑色，就這樣形成我們的 Ground Truth，將每一組的照片數量加在一起，少說也有一千多張(下圖為利用繪圖軟體標記的照片)

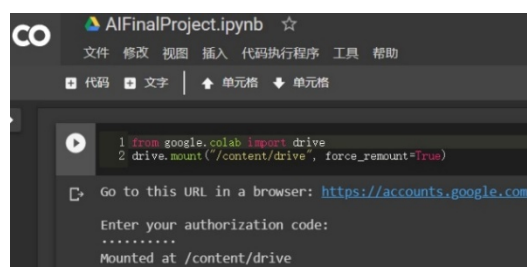


訓練 Model 環境

這次因為照片向量的維度也不小，我們這組的筆電幾乎都沒有 GPU 加速，所以我們後來找到了 Colab，他是 Google 提供給想要做機器學習或是深度學習的工程師一個不錯的環境，裡面有免費的 GPU 加速環境，對於訓練 Model 的速度提升很多。

這個環境非常方便，可以讓我們將資料都丟到 Google 的 Server 上去跑，雖然還是有他的上限，但處理一般的訓練應該是綽綽有餘。

(下圖為 Colab 環境截圖)

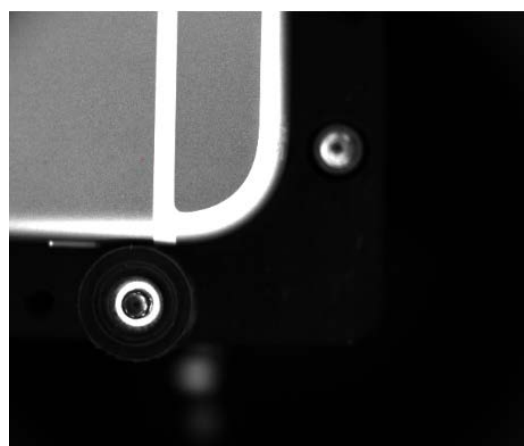


具體作法(Method)

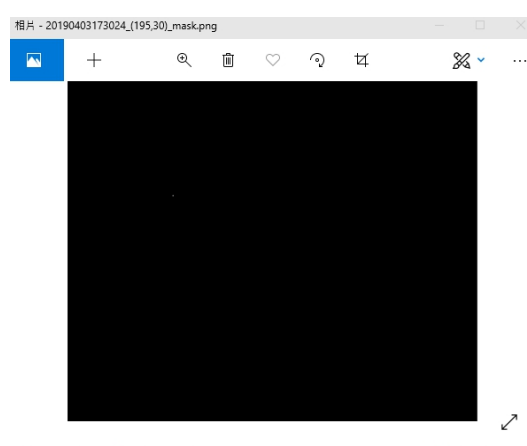
資料前處理(Data Preprocessing):

首先我們發現 ground truth 有許多全黑的照片(表示完全沒有刮痕)，這樣對於訓練來說，會多了很多對判斷刮痕沒有幫助的照片，所以我們這一組每個人分別看一部份 ground truth，將刮痕較明顯的照片額外抓出來(大約五百筆)，利用這五百筆來訓練我們的 model。

(下圖為沒刮痕的手機原圖)



(下圖為上圖沒刮痕的 ground truth)



第二我們將照片用灰階(grayscale)
讀入當成 Input

(下圖為 Demo 灰階效果)

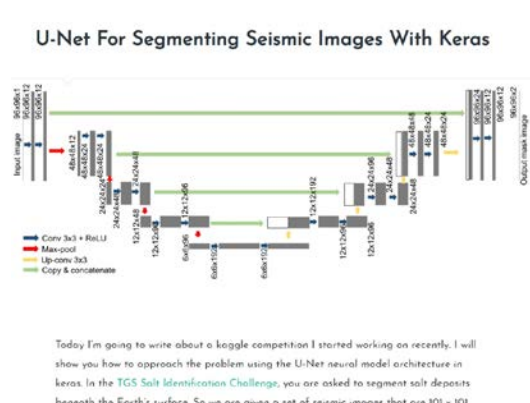


第三我們將照片 Resize 到
128*128，這麼做其一是為了統一照片格
式，其二是為了避免刮痕佔整張照片太小
的一部份，所以我們做了 resize。

(下圖為 resize 的 code 截圖)

```
img = load_img(path + 'images/' + id, grayscale=True)
x_img = img_to_array(img)
x_img = resize(x_img, (128, 128, 1), mode='constant', preserve_range=True)
```

模型挑選(Model Selection)



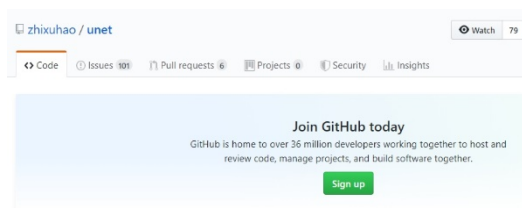
(上圖為 Demo Unet 的基本架構)

由上圖我們可以看到這次我們實作
image segmentation 是參考自 Unet 的
Network，他是先做 convolution 抽取出
相片的特徵後，在一步一步
Deconvolution 回去原圖。

方法一：

我們 Google 搜尋 keras unet，看到
幾個 github 和幾個網頁，我們這組每個
人分配一下，分別看不同的 model，因為
無法直接看出 model 是否適合，我們分別
實作出一種 model，看刮痕辨識是否有效
果。

其中一個 github 是



(上圖是我們試的第一個 github)

```
from model import *
from data import *

#os.environ["CUDA_VISIBLE_DEVICES"] = "0"

data_gen_args = dict(rotation_range=0.2,
                      width_shift_range=0.05,
                      height_shift_range=0.05,
                      shear_range=0.05,
                      zoom_range=0.05,
                      horizontal_flip=True,
                      fill_mode='nearest')

myGene = trainGenerator(2, 'data/membrane/train', 'image', 'label', data_gen_args, save_to_dir = None)

model = unet()
model_checkpoint = ModelCheckpoint('unet_membrane.hdf5', monitor='loss', verbose=1, save_best_only=True)
model.fit_generator(myGene, steps_per_epoch=300, epochs=1, callbacks=[model_checkpoint])

testGene = testGenerator("data/membrane/test")
results = model.predict_generator(testGene, 30, verbose=1)
saveResult("data/membrane/test", results)
```

(上圖為主程式的截圖)

利用 generator 將資料分批讀進來
，可以避免資料量過大，也有使用 check
point 將最好的參數儲存進去，
雖然 accuracy 很高，但我們將 model 預
測的刮痕圖輸出，發現這個方法預測都是
黑色，因為他學習到如果都猜黑色可以將
loss 壓到最低，我們調整模型的參數和
filter 的個數，效果還是沒有很好，所
以我們放棄了這個方法，我們繼續試了下一
個方法。

方法二：

我們繼續看第二個 github

keras_segmentation	added support for finetuning
sample_images	readme updated
scripts	minor changes
test	added cli tool
githubignore	a
LICENSE.txt	added support for finetuning
README.md	Update README.md
requirements.txt	now python3 compatible as well
setup.cfg	added support for finetuning
setup.py	added support for finetuning
README.md	

Image Segmentation Keras : Implementation of Segnet, UNet, PSPNet and other models in Keras.

第二個 Model 是 vgg_unet，這個模型是先是
利用 pretrained 的 VGG 網絡當 encoder 將
照片做特徵擷取，之後再用
UpsamplingConvolutional 2D，轉換回原圖大
小，輸入數據後，在訓練過程中，精準度一直

都非常高，經過前面的經驗，也是預測出全部黑的畫面，但是可以預測出一些點出來，但是比例還是非常得小，原因也是因為如果全部猜黑色精準度還是最高，這在我對於這種主題的了解知道，盡量不要用太大的 pretrained model 去做，因為要分類的 class 只有一個就是刮痕，但是形狀也是有很多形狀。

方法三：

直到我們看到第三個網頁，這次的（上圖為該網頁作者的介紹該 model 的截圖）

原本這個 model 是用來參加 kaggle(國際非常知名的資料科學探勘比賽網站)TGS salt 岩層地質分析比賽，我們觀察發現雖然和前兩個 github 一樣都是實做出 Unet，但是這個 filter 的 channel 數相對少很多，大概是從 16 開始慢慢到 128 而已，但前面幾個都到快 1024 了，因為我們的問題可能不需要將 channel 數量拉到太多。

每層都加 Dropout，並且利用 Batch Normalization，可以讓我們預測出刮痕。

模型評估與實驗結果分析

(一)(Experiments & Results)

模型評估(利用 IOU):

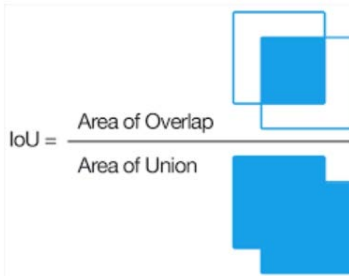
這是助教要求我們評分的方式，他的概念是計算 label 和 predict 的白色計

Final Project 才看到一線希望，我們下面會詳細介紹這個方法，

Today I'm going to write about a kaggle competition I started working on recently. I will show you how to approach the problem using the U-Net neural model architecture in keras. In the [TGS Salt Identification Challenge](#), you are asked to segment salt deposits beneath the Earth's surface. So we are given a set of seismic images that are 101 x 101 pixels each and each pixel is classified as either salt or sediment. The goal of the competition is to segment regions that contain salt. A seismic image is produced from imaging the reflection coming from rock boundaries. The seismic image shows the boundaries between different rock types. Lets look at some of the images and the labels now.

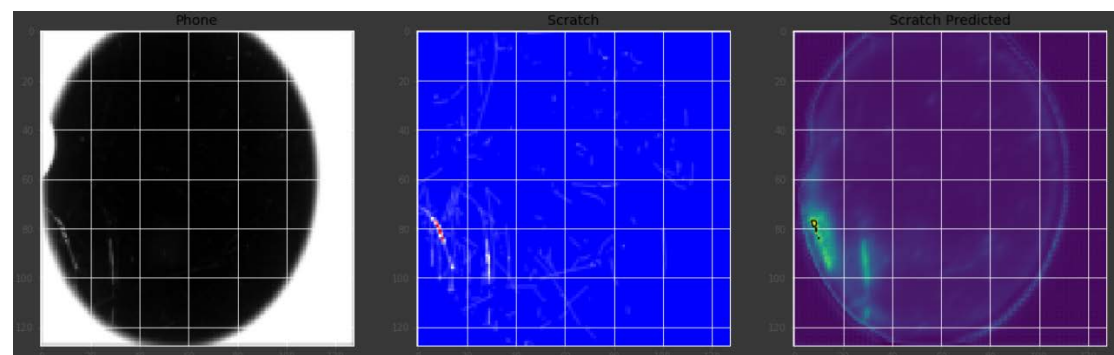
```
In [1]:
import os
import random
import pandas as pd
import numpy as np
```

算，如下圖的公式

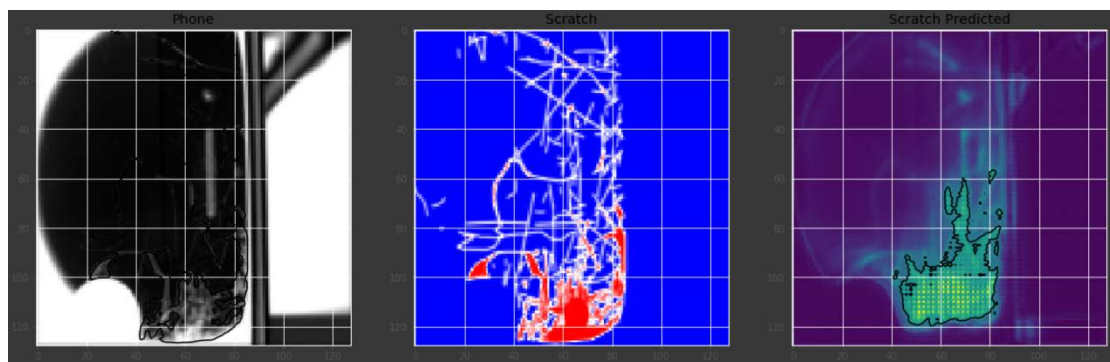

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

我們這一組後來有大概看一下我們分出來的結果，以下我們 Demo 三組照片，分別是下一頁的圖一到圖三，最左邊的是原圖，中間那張背景藍色的是人為標記的 ground truth，最左邊那張背景為紫色的是我們預測出來的結果，由圖其實我們可以知道我們的模型其實有預測出刮痕，但是因為其實很多照片有些主別是用程式軟體標 label，有些人是用繪圖軟體，這樣導致 model 會無法確認到底是要分辨哪一種。

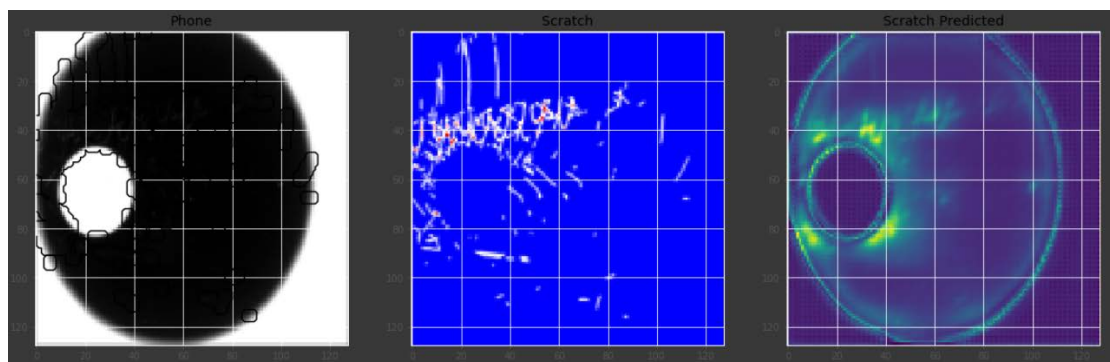
下圖為圖一(I0U=0.32)



下圖為圖二(IOU=0.38)



下圖為圖三(IOU=0.24)



模型評估與實驗結果分析(二)

模型評估(準確率)

利用 keras 的 accuracy 來計算準確度，

```
Epoch 00099: val_loss improved from 0.06143 to 0.06132, saving model to model-tgs-salt.h5
Epoch 100/100
324/324 [=====] - 3s 9ms/step - loss: 0.0612 - acc: 0.8187 - val_loss: 0.0613 - val_acc: 0.7268
Epoch 00100: val_loss did not improve from 0.06132
```

這個精確度雖然沒有到很高，但是我們覺得這個模型其實是有學到東西的(代表模型沒有全部猜黑，他有預測他認為是刮痕的地方)。

未來期望與改進

沒聽過大家的報告後，我們發現其實還有改進的空間，我們應改可以結合影像處理的技術，加上 CNN 提取特徵的方式將 Final Project 處理得更好。

心得感想(Discussion)

F74066022(徐澤淞)

這次 Final Project 真的算是非常難，在寫第一次的數字辨識時，還覺得可以接受這個難度(但是也花一段時間熟悉 tensorflow)。

對於數字辨識這個專案，我們就需要對照片進行前處理，這是以前都沒有接觸過的一個領域，影像處理老師上課也講過，印象最深刻的就是對於電腦來說，其實一張照片是三層 channel(分別代表 RGB)來儲存，我們如果要處理的跟彩色無關，可以將照片改成灰階讀取，並可以根據想要的 pixel 數，對照片 Resize。

接著就是 tensorflow 這個寫深度學習的套件，用起來還是很吃力，因為畢竟是別人寫的軟體，將資料平均打亂，一次只讀一點進來 train，畢竟 GPU 的容量有限，也花了一段時間理解助教的範例 code，助教也都很耐心的教導我一些不懂的地方，和 tensorflow 的特殊用法。

到了這次 Final Project 有鑑於模型變得比較複雜，我們採取 Keras 這個比較簡潔的深度學習套件(適合初學者使用)，一開始非常挫折，因為我們用 Unettrain 出來的 model 預測都是全黑的，這樣其實完全沒有解決問題，我們參數調了很久都沒有太大的幫助。

最後我們繼續上網找資料，終於找到一個是別人 kaggle 時做過的 unet，有真正處理過 data 的模型架構，和分析的方法，觀察後發現原來我們資料前處理 resize 和灰階很重要，而且其實 model 的 filter channel 數量不需要太大。

有了這次經驗其實我們往後對於處理

問題有了 SOP(標準作業流程)，應該可以上 kaggle 這個國際知名的資料探勘比賽，找有沒有相關的比賽 data 和我們要時做的專案相同，因為上面有很多高手，可以跟他們學習，筆自己在家閉門造車還要強很多，不只可以學到資料前處理的方法，更可以學到建立模型參數調整的方法，這些都是別人寶貴的經驗，他們無私的放在網路上給大家學習，我們應該要向高手多多學習。

上了一學期的人工智慧課程，從一開始完全不會打 python，查資料練習 python，把 python 一些基本的用法的研究清楚，就花了一段時間，研究完 python 後，要花不短的時間研究 tensorflow 和 keras，現在對於這些程式套件已經有一點了解了。老師也將我們的基本功教的很扎實，因為期中考必須精通老師的講義內容，我和同學花一個假日，討論了老師上課的內容，對於 propagation 參數更新很熟悉了，甚至是後來教的 RNN，看似一張圖但其實是一個 neural network，這些都是人工智慧的基本觀念，有了這些相信以後可以繼續深造這方面的能力，畢竟人工智慧是現在這個時代的一個很有用的工具，要將他用在哪裡，就取決於用的那個人。

F74062044(黃政璋)

這學期的課真的使我獲益匪淺，雖然我身為一位資訊系學生，但是修這堂課前是完全沒有碰過 python 的，因此第一次作業要畫 activation function 時，花了相當多時間了解這語言的語法，第一次作業要做數字辨識，也花了相當多時間研究

如何使用那些 API 與 tensorflow。

事實上我並不是很認同許多人說老師上課不好，內容安排不佳，老師教我們的確就應該是這些深度學習的觀念，要打出 code，上網 google 都有相當多影片與資源，因此我覺得老師著重這些觀念是很好的，的確這些內容都很廣，有跨到影像、聲音、甚至文字等，難免學生會吸收不良，但是老師點到最核心的觀念其實就很夠了，其餘知識我們更應該去自主學習，一方面能更加弄懂老師上課的內容，一方面也能加強自己上網學習的能力，抱怨內容過難時，事實上應該也要檢討自己花了多少時間在這堂課，等盡一己之力再來抱怨。

最後的 FinalProject 真的相當有趣，這可以說是我大學以來遇到最有挑戰的一個作業，從一開始完全不知道運用這些 data，到後來不斷的嘗試、不斷的收集資料，最終好不容易才看得到圖的成果，雖然我們 IoU 才接近 0.4 而已，但對我們這組來說，已經是莫大的成就了，因為我們對影像處理的技巧相當有限，但我們還是憑藉著深度學習的威力，硬是給他做了出來，唯一有點小小建議，就是希望 FinalProject 可以統一早點給好資料，並做好標記，難免會遇到有人亂標記甚至亂傳，這造成我們很多困擾，我們前面就花了數十小時在那邊挑選資料與標記，以至於真的花時間在嘗試的模型時，被壓縮了相當多，如果有更充裕的時間，我相信我們這組應該能在這僅有的 data 與資源做出超過 IoU 0.5 的結果。

另外建議老師可以鼓勵大家多使用 Colab 的資源，事實上我們幾乎沒用過 Nvidia 的資源，因為還要重新架環境跟弄資料，其實也需要花相當多時間，而 Colab 可以隨時使用，而加速效果也非常

的好了，因此可以多鼓勵同學多使用 Colab 來 train 模型。

最後真的很謝謝兩位助教跟老師總是不厭其煩的回答我們問題，有時我跟我同學總會想弄清楚背後的原理，不斷的煩助教、問助教，難免覺得不好意思，但是助教總是很認真的講解甚至畫圖給我們看，真的很令我們感動。老師對這塊的熱忱，真的深深打動了我，如果有機會我真的還想繼續碰這塊，讓自己在這個領域能有更大的精進，最後再謝謝老師，盡心盡力的教會我們，請非常好的講師講課給我們，讓我們學到不只是人工智慧這塊，而是更廣的知識，讓自己上完這堂可以滿載而歸，也很驕傲的告訴大家，我們上完這堂課，是可以獨當一面寫出 tensorflow 與 keras 來解決一個實際的問題，讓自己未來能多一項才能。

F74064054(戴宏諺)

剛修這堂課後，一開始覺得這堂課很有趣，因為教的東西是比較算是基礎的內容，真的非常喜歡。在第一次作業覺得我們馬上看論文有點太早了，因為我們都沒辦法實作，所以讀起來是很抽象的，比較沒有感覺，希望可以經過第一次實作作業後，讀一篇論文，這樣會比較有感覺。

覺得跟老師說有很多實作有很大的出入，目前為止也只有做出兩個實作(數字辨識和期末專案)，覺得應該多要求些課堂作業，或是程式作業，讓學生可以多多思考，因為如果每次去上課都是聽聽講之後，就沒有人之後還會去讀他了，之後老師的課程比較多是，各個領域都教一點，讓學生覺得很難將課程內容連成一條線，也就很難融會貫通，不過這也很為難老師，畢竟這是一個技術門檻較高的領域，對於學生來說，相對也是比較吃力的。

在第一次作業，覺得其實算是簡單的，因為網路上都有很多的教學資源，光是看官方文件做出來，其實也花兩天的時間而已，對於數字辨識的分類器，是讓我開始了解深度學習的起點，也讓我知道這是需要透過程式碼實作才能知道的，不過應該也是因為這份作業期限是一個月，讓課程進度可能就被這個作業拖累了，個人認為功課最多兩周是最好的時間。

在助教教學 tensorflow 方面，其實教得跟沒教是一樣，因為同學都沒有任何概念，不管是如何利用 Pipeline 輸入數據或是如何使用一些機制讓數據龐大的資料可以批量輸入，這也是這堂課的難處，建議在期初就先跟同學先去研究深度學習的基本概念，以及利用 tensorflow 做出一些很簡單的分類器，進來上課後，老師教課也可以教得比較輕鬆，不太需要擔心同學聽不懂。以及多出些其他的分類器，或是模型實作，一定能更讓同學了解 tensorflow。

在老師請 Nvidia 講師來講課是覺得非常得好的，因為以往這種課程其實都是非常需要計算資源，也提供了非常好的環境給我們做訓練，不過唯一可惜的是，講師來講課的模型是偏向自然語言的領域，其實一開始聽起來也非常得陌生，做出來也是非常陌生，不過能聽過一些其他領域的方法後，其實也可以知道自然領域也算是利用類似卷積的方式去掃過全部的文字。

最後在做 Final Project，一開始是真的完全不知道要怎麼做，也完全不知道要怎麼利用 tensorflow 去作出 unet 或者是一些 Fully Convolutional Network，最後還是在網路上一些大神提供一些已經寫好的 model，但是對於自己用別人寫好的 model 去改，是有點不甘

心的，因為其實我起初是想要自己建立一些模型來去辨識這些資料，這也跟老師說，如果你們看到一個新的模型，能不能自己做一個出來呢，這真的比較難一點，不過我們這組在最後有訓練出來一些情況是真的非常開心。

最後要謝謝老師和兩位助教每個禮拜都提供很好的課程給我們學習，最後期末報告的氣氛也沒有嚴肅，也覺得老師人非常好，真是感謝你們。