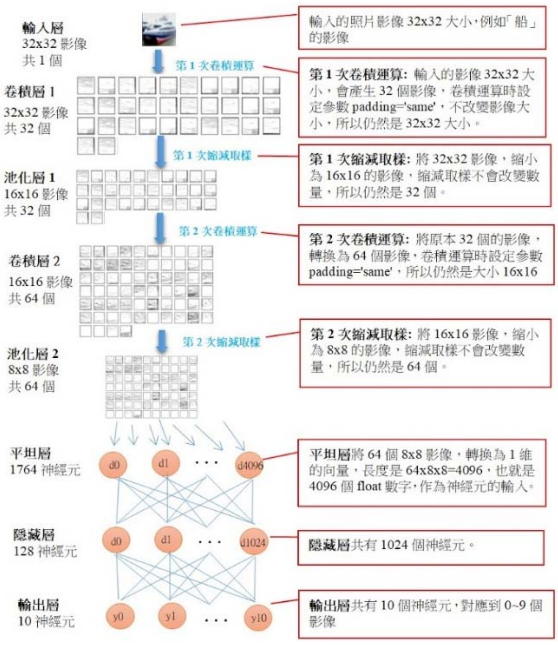
Deep Classification

這次的作業，我試了三個 model 去進行預測：

	描述/思路	改良/心得
(53%) CNN_basic	原本是用來預測 cifar10 的 model (71%原精準度) 容易訓練、易懂好寫是它的特色， 是我一開始作業練習使用的 model。	參考用書： TensorFlow+Keras 深度學習人工智慧實務應用
(50%) Mnist	原本是用來預測手寫數字的 model (99%原精準度) 我就想著也許我也可以將 data 灰階化(黑白化)， (也許黑白化的資料，輪廓會比較好抓!?)	灰階圖流失了許多資訊 仍然擁有 50%的精確度
(54%) AlexNet	1. 將 input data 轉成 ndarray(227,227,3) 2. 右手的資料，將其左右翻轉 (思路: 讓右手的 data 跟左手的 data 角度一致化) 3. 使用 AlexNet 進行訓練 4. 訓練過程使用的資料有做 shuffle (訓練資料分布較為均勻)	(227,227,3)的資料略大 於是改用批次讀取、訓練

CNN_basic (53%)



〈CNN 組成〉

Conv2D(卷積層)、MaxPooling2D(池化層)、Flatten(隱藏層)、Dense(隱藏層)

〈此 model 特色〉

在 cifar10 (keras 的 dataset)裡面預測，可以有 71%的精準度，此 model 簡易、容易訓練(訓練時間短、而且可以同時將所有資料都塞進去訓練，不需要做分批讀取)，因此我一開始選擇此 model 作為此作業練習時的第一個 code，用來熟悉 CNN 的架構。

過程中曾經遇到的問題：

1. input data 預處理
   - I. 1920*1080 的圖是必須縮小處理，直接餵給任何一個 model，都會太佔空間。
   - II. 我使用的處理方式是將圖形縮小成 (32*32, CNN_basic 使用)、(227*227, alexNet 使用)，轉成(227, 227, 3)的 ndarray 之後，將其儲存起來，之後訓練要使用時，在把它 load 到 memory 裡面。
   - III. 網路上有著許多 CNN、alexNet 的 code，但是各家版本使用的 input 格式都不盡相同，我之前就遇到 input 格式是 tensor，在資料轉換上，會有一些問題(如附圖)

```
In [65]: from keras.layers.convolutional import Conv2D
         conv_1 = Convolution2D(96, 11, 11, subsample=(4, 4), activation='relu',
                                name='conv_1')(inputs)

/home/smilewater/anaconda3/envs/tensorflow-gpu/lib/python3.6/site-packages/ipykernel_launcher.py:3: UserWarning: Update your `Conv2D` call to the Keras 2 API: `Conv2D(96, (11, 11), activation="relu", name="conv_1", strides=(4, 4))`
  This is separate from the ipykernel package so we can avoid doing imports until

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/home/smilewater/anaconda3/envs/tensorflow-gpu/lib/python3.6/site-packages/keras/engine/topology.py in assert_input_compatibility(self, inputs)
    424             try:
--> 425                 K.is_keras_tensor(x)
    426             except ValueError:

/home/smilewater/anaconda3/envs/tensorflow-gpu/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py in is_keras_tensor(x)
    402                             tf.SparseTensor)):
--> 403         raise ValueError('Unexpectedly found an instance of type `' + str(type(x)) + '`. '
    404                          'Expected a symbolic tensor instance.')

ValueError: Unexpectedly found an instance of type `<class 'numpy.ndarray'>`. Expected a symbolic tensor instance.
```

2. ResourceExhaustedError

   我原本很擔心是自己電腦的 memory 不夠用，需要加買記憶體，不過後來這問題在電腦重新開機之後，神奇地迎刃而解。原本跑不動的 model，在重開電腦後，跑得動了!

   (結語:遇到 ResourceExhaustedError 別灰心，也許是有些暫存記憶體還沒有釋放出來，重新開機有機會有奇效!)

```
ResourceExhaustedError: OOM when allocating tensor of shape [3,3,256,384] and type float
	 [[Node: training_5/SGD/Const_4 = Const[dtype=DT_FLOAT, value=Tensor<type: float shape: [3,3,256,384] values: [[[0 0
 0]]]...>, _device="/job:localhost/replica:0/task:0/gpu:0"]()]]

During handling of the above exception, another exception occurred:

ResourceExhaustedError                    Traceback (most recent call last)
<ipython-input-106-463a8007a95a> in <module>()
      1 train_history=model.fit(x_img_train_normalize, y_label_train_OneHot,
      2                         validation_split=0.2,
----> 3                         epochs=10, batch_size=1, verbose=1)

/home/smilewater/anaconda3/envs/tensorflow-gpu/lib/python3.6/site-packages/keras/models.py in fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight, sample_weight, initial_epoch, **kwargs)
    865                               class_weight=class_weight,
    866                               sample_weight=sample_weight,
--> 867                               initial_epoch=initial_epoch)
    868
    869     def evaluate(self, x, y, batch_size=32, verbose=1,
```

想要詢問助教的問題+project 建議

1. 這次 project 裡面的 label 分布頗為不平均，如果能將每一個 label 有著差不多的資料數量，是否可以有比較好的訓練效果呢？

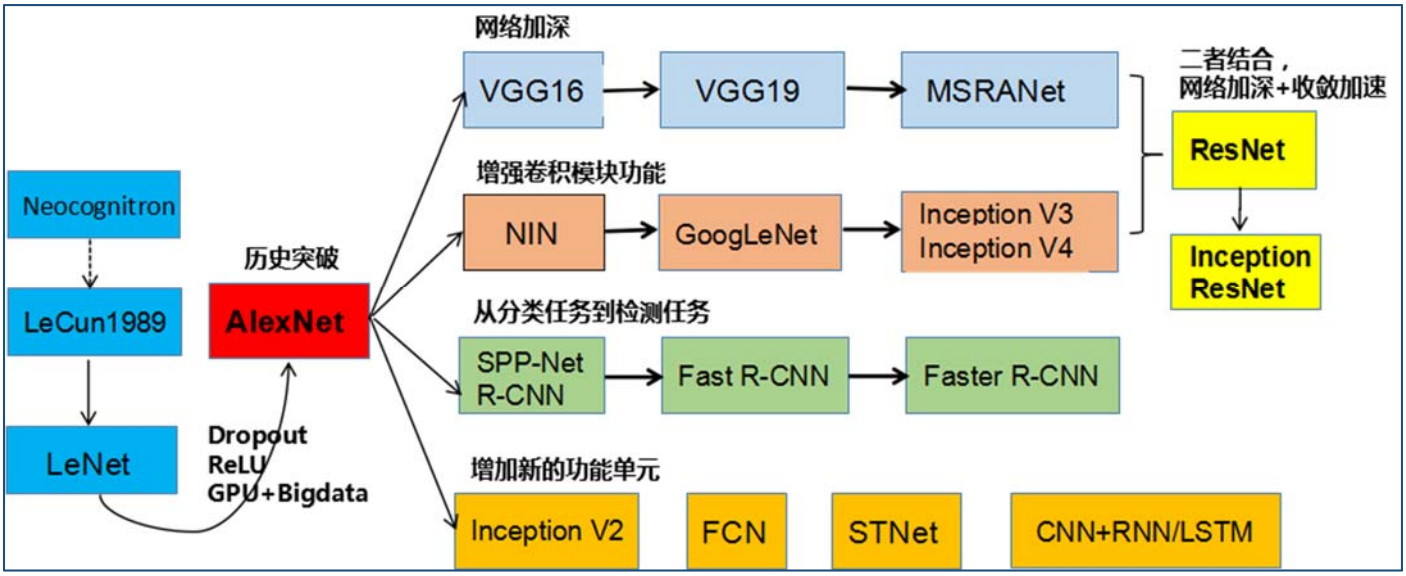
   (某些 label 出現次數極少)

寫報告時的發現：

jupyter notebook 上傳到 github 的形式非常好看，完整呈現訓練與寫程式的過程，同時也可以加上自己的小標題與註解，讓讀者更加容易可以理解本程式的功用。

心得：

因為自己在 deep learning 這個領域剛起步，中間經常感到迷惘與困惑，
在學習的過程中，我不停地嘗試，從 data 的存取、轉換，以及去查 model 的建構方式，
感覺到這對於學習 keras 不到 1 個月的我來說，是個很大的挑戰，
但也在這過程中，學習了如何去統整資料，以及去把自己想要問的問題具體化，
當努力了許久，將 AlexNet 拼湊出來時，那種興奮是不言而喻的，
做了 deep learning 的實作，可以讓自己更加理解老師上課講的內容，以及去享受腦力激盪的過程。



AlexNet 是一個歷史上的突破，在這樣的 model 上進行訓練，我深深地感受到它的強大之處，以及更好奇的是，作者是怎麼想出這樣設計的呢？

特別感謝

(助教) liao andrew：花了許多時間跟學生討論此次 project

(網站) Stack Overflow：迷茫之時，到這個網站可以找出一條明路。

(循著前人的經驗，與解決問題的方法，往前邁進)

(課程) Share Course (Python 資料科學實作)：教會我使用 anaconda，並在上面進行 python 實作。

(書籍) Tensorflow+Keras 深度學習人工智慧實務應用：

提供基本知識，並且以簡單易懂的範例來讓初學者容易上手。