Image (cifar10) Classification

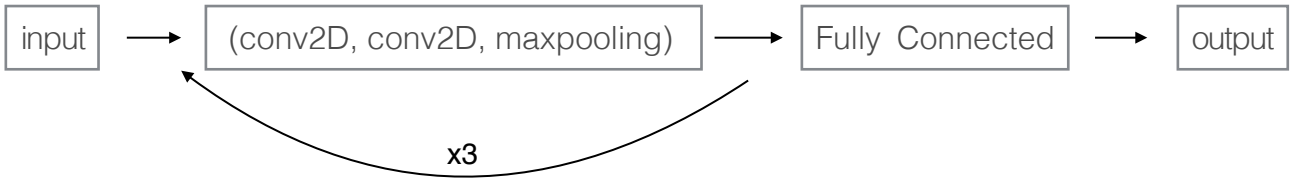
R05942078，陳偉寧

1. Supervised Learning

以5000筆labeled data做supervised learning, 我選擇使用CNN作為我的classifier。實作上我主要使用keras這個deep learning framework，並使用GPU作為加速的運算資源。以下將詳細介紹我CNN的Layer與其performance。

A. Convolutional Neural Network

我的架構和傳統CNN差異不大：



嘗試多次的結果後，我發現convolution network做三次，fully connected network 放三層hidden layer (with 1024, 1024, 2048 neurons)時效果較好。另外，我的其他參數分別設定如下：
dropout: 0.5, conv dimension: (64,64) (128,128) (256,256) (每輪做兩次，做三輪), pooling_size: (3,3)
optimizer: SGD。以下將簡述對於不同structure, parameters所得到的performance。

B. Network Structure vs Performance

Convolution 2D dimension (three times, with each 2 conv2D per time):				
acc \ conv2D dim	(32, 64, 64)	(64,128,64)	(64,128,256)	(128,256,128)
public	0.665	0.702	0.7118	0.7176
private	0.670	0.694	0.7272	0.7310

上表皆為固定fully connected layer 為 2 hidden layers with (1024, 1024) neurons 所得到的結果。我最後選用(64, 128, 256)的架構，主要是在做self-train的時間考量。

FC network hidden neurons (multiple layers):					
acc \ FC layers	(512,1024)	(1024,1024)	1024 x 4	(1024,1024,2048)	2048 x 5
public	0.665	0.6956	0.699	0.7044	0.702
private	0.670	0.7096	0.7116	0.7054	0.694

上表皆為固定conv2D架構為(64,128,64)情況下所得到的結果。最後我選用了(1024,1024,2048)的結構。

2. Semi-supervised Learning: self-training

Self-training 是這次三個方法中表現最好的，並且最佳的結果在public與private set上分別可以達到0.78及0.7952的正確率。以下將介紹我的實作方式。

A. Implementation

1. Initial

首先用5000筆labeled data train一個model，並用此model predict 全部的 unlabeled data。

2. Iterative reload unlabeled data

接著再從這些被標記的data中選出scores最高的5000筆，和原先的labeled data concatenate在一起，重新train一個新的model。

3. Dropout picked data

每次concatenate一些新的unlabeled data，我們就將它從原先的unlabeled data set去掉，以確保unlabeled data不會被重複使用。

重複步驟3、4，直到所有的unlabeled data都被用完為止。另外，在一開始load unlabeled data時，我也將test data一併concatenate進去，因此共有55000筆unlabeled data。

B. Performance

這裡主要探討一個問題：增加unlabeled data真的對self-training有幫助嗎？這也是我最想探討的問題。因此這邊比較了在每次reload unlabeled data後對performance 到底有多少影響。

由上表我們可以發現增加unlabeled data個數對performance有顯著的影響。

Number of unlabeled data									
acc	0	6000	12000	18000	28000	34000	40000	45000	55000
public	0.716	0.721	0.7274	0.7376	0.7658	0.7640	0.7780	0.7764	0.7798
private	0.7276	0.7398	0.7406	0.7504	0.7678	0.7816	0.7868	0.7938	0.7952

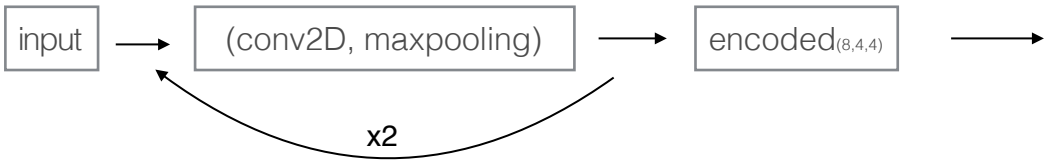
3. Semi-supervised Learning: autoencoder, SVM

第二種semi-supervised的implement我使用了autoencoder作為feature extractor，並以SVM作為最後的classifier。SVM雖然運算量相較於CNN少很多，但performance非常糟，因此我也著墨較少在這部分。

A. Implementation

(1) convolutional autoencoder

這邊一樣是使用keras實作，其中我用了兩層的 conv2D/maxpooling，如圖。



最後的output shape是(8,4,4)，換言之我們將原先的3072維data map到256維。

(2) Support Vector Machine

SVM使用的是sklearn的tool，並且選用one-vs-one, RBF kernel。

B. Performance

這個方法影響accuracy的一個因素是SVM的punishment coefficient C。我嘗試了C從0.1到500，發現當C=20時performance 最好，accuracy有0.51。

4. Discussion

以下簡述幾點這次實作中所觀察到的技巧\現象。

1. 由於NN對input range很敏感，因此要記得加BatchNormalization()，前幾次沒有家發現對performance影響非常大。
2. optimizer方面，原先使用Adam收斂很快，但performance遠不如SGD。使用SGD即使是只用labeled data也可以有超過70%的準確率。
3. 由第一題的數據可知，CNN後面的FC network對performance影響其實沒有很大，反而是前面convolution 2D的dimension比較重要。
4. self-learning當中unlabeled data確實對performance有顯著的影響。
5. 最後的acc一直在大約79%，要在更好的話，可能需要用到一些clustering方式得到一些額外的informaion。