

Machine Learning HW3 Report

b02901122 電機四 劉致廷

1. Supervised-learning:

我使用的是 Keras toolkit, backend 為 tensorflow, 在 supervised-learning 中, 我使用 CNN 作為我的 Model, 以下介紹我使用的 model:

```
Input(32,32,3)→Relu(Conv(32,3,3))→Relu(Conv(32,3,3))→Maxpooling→Relu(Conv(64,3,3))→Relu(Conv(64,3,3))→Maxpooling→  
Relu(Dense(512))→Softmax(Dense(10)) loss: categorical_crossentropy, optimizer: Adam
```

5000 筆的 data 部分, 一開始我並沒有做什麼前處理, 只有切 500 筆當作 validation, batch_size=128, epoch=40, 唯一的變化是是否多加一層的 `Relu(Conv(128,3,3))→Relu(Conv(128,3,3))→Maxpooling`, 之後會以兩層或三層的 model 來代表。第一次實際測出的 validation data performance 為

兩層: acc = 0.5518, 三層: acc = 0.585

接下來, 便嘗試使用 keras 內建的 `Imagegenerator`, 只使用其中的 `weight shift`, `height shift`, `horizontal flip`, 在一樣的測試下, validation data 的 performance 為

兩層: acc = 0.6700, 三層: acc = 0.680

因此, 可以推斷, 加上 Imagegenerator 與疊成三層的表現都會比較好。

2. Semi-supervised learning(1):

我使用的方法是 Imagegenerator + Selftraining, 我使用的 self-training 指的是先利用 label data (4500train, 500vali) train 出一個模型後, 接著去標記 unlabel data, 取其中較有信心的 data (probability > 0.99) 加入舊有的 label data 中, 接著利用合併的 data 繼續 train 我的 model, 經過一定的 Epoch 數後, 便重新 train "label data", 也就是只有繼續 fit 4500 筆資料, 目的是因為害怕在 label 後 retrain 的過程因為標記錯誤而讓 model 偏移, 因此在利用絕對正確的 label data 做一些修正, 接著就繼續 label unlabel data, retrain, train 舊有 data...。經過幾次的循環, 便完成最終版本的 model。

在過程中, 我用了許多變因來測試我 model 的 performance, 例如改變循環數目, model 的複雜度, 是否加入 batchnormalization, activation function 的不同, 是否加入 test data 來 self-training, 甚至在加入 unlabel data 的時候是 weight by confidence 而不是只看 0.99, 這些會在第四題做討論, 以下會以 performance 最好的 model 說明。

這是我最終版本的 model:

```
INPUT(32,32,3)→ELU(NORMAL(CONV(64,3,3)))→ELU(NORMAL(CONV(64,3,3)))→MAXPOOLING(2,2)→DROPOUT(0.25)  
ELU(NORMAL(CONV(128,3,3)))→ELU(NORMAL(CONV(128,3,3)))→MAXPOOLING(2,2)→DROPOUT(0.25)  
ELU(NORMAL(CONV(256,3,3)))→ELU(NORMAL(CONV(256,3,3)))→MAXPOOLING(2,2)→DROPOUT(0.25)  
ELU(NORMAL(DENSE(512)))→DROPOUT(0.5)→ ELU(NORMAL(DENSE(128)))→DROPOUT(0.5)→SOFTMAX(DENSE(10))  
LOSS: CATEGORICAL_CROSSENTROPY, OPTIMIZER: ADAM (LR=0.001)
```

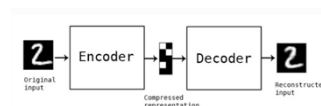
我 channel 的架構為 64, 128, 256 (三層), 使用 BatchNormalization 來 normalize 每一層的 filter 與

Dense, activation 的部分, 我使用了比較特殊的 "elu", 使其不會歸零 input 小於 0 的值,

另外我在 training 的過程中, 加入了 (1) `Early Stop` 與 (2) `ModelCheckpoint`, 使其比較不會 overfit 並且可以存下 Val_acc 最高時的 model, 循環的 iteration 我設為 6, 也就是 "train Lable, predict, train confidence data" 循環六次, confidence 的部分我只提取了機率 > 0.99 的 data。在這樣的 model 底下我得到 Val_acc = 0.862, 而最終上傳到 kaggle 上的 performance 為 **Acc = 0.847 !!**

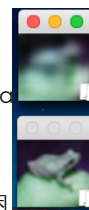


3. Semi-supervised learning(2):



我使用的方法是 autoencoder+selftraining, 所謂的 autoencoder 如圖所示, 利用一個 encoder 與 decoder 的 network, 可以產生維度比較少的 vector 來代表 input 的圖片, 我使用 128 維的 vector 來代表圖片的 feature, 接著嘗試了兩種方法(1.) SVM (2.) 建構一個 DNN 來 train feature 並且使用 2.所述 selftraining. SVM 的方法為利用 label 的 feature 建 SVM model, 並且 label unlabel data, 接著拿 label 與 unlabel data, 重新建構 SVM 後 label test data, 最後 performance 只有 **Acc=0.39**. 方法二則是使用一個 DNN, 所有的 data 都通過 encoder 產生 feature, 接著就是 “train label,predict,train confidence data “做循環, 而最後的 performance 則是有稍微的進步,

Acc = 0.47, 我認為最大的問題是 autoencoder 其實是非常困難 train 的好, 如果一開始就有偏差, 我想用什麼後續的方法都蠻困難能正確分類。如右圖是 autoencoder 的 input(下) 與 output(上)。



4. Compare and analyze your result:

我目前只有對 2.的方法做分析與比較, 因為用此方法的正確率比較高, 且較可以比較與改善。首先, 我比較是否有加入 weight by confidence 會比較好, 而其他的變因為層數, 模型複雜程度, 與 data /255。

型態	模型	Validation Acc	模型	Validation Acc
No weight by confidence	2layer,32-64 模型, no /255	0.76	3layer,64-128-256 模型, no /255	0.766
	3layer,32-64-128 模型, no /255	0.74	3layer,64-128-256 模型, /255	0.688
Weight by confidence	2layer,32-64 模型, no /255	0.662	3layer,64-128-256 模型, no /255	0.712
	3layer,32-64-128 模型, no /255	0.696		

推測: No weight by confidence, 64-128-256 模型比較好 接下來比較是否加入 test data 來一起 label 並且 retrain 會比較好

型態	模型	Validation Acc
No test data	64-128-256 模型, no /255	0.766
	64-128-256 模型, /255	0.678
Add test data	64-128-256 模型, no /255	0.784
	3layer,64-128-256 模型, /255	0.680

推測: Add test data, 不要 /255 比較好

最後綜合比較是否加 normalization, 換 elu, 甚至在 DNN 的地方再加一層 128 個 neuron.

型態	模型	Validation Acc
No add 128	No normalize,Relu	0.80
	Normalize,Relu	0.824
	Normalize,elu	0.832
Add 128	Noemalize,elu	0.862
	Noemalize,elu,/255	0.862

既然/255 有時會比較好有時差不多, 那我就統一不要/255, 並且就以 **validation 0.862** 的模型當作我的最終版本。