Machine learning hw3

B03902071 葉奕廷

1.Supervised learning

我首先使用了 CNN 來做 supervised learning。 利用 CNN 做 cifar-10 辨識做得很好的例子在網路上可以找到很多，但是因為我們的 labeled data 很少，所以若是參數太多雖然在原始 cifar-10 上可能結果出眾，在只有 5000 筆 labeled data 的情況下可能就會有 overfitting，所以我並未讓 supervised learning 的 CNN 長得太深。 我試了兩種CNN， model A 與 model B。

layer	modelA
1	Convolution, channel=32, filter=3x3, relu
2	Dropout, rate=0.2
3	Convolution, channel=32, filter=3x3, relu
4	MaxPooling, kernel=2x2
5	Convolution, channel=64, filter=3x3, relu
6	Dropout, rate=0.2
7	Convolution, channel=64, filter=3x3, relu
8	MaxPooling, kernel=2x2
9	Convolution, channel=128, filter=3x3, relu
10	Dropout, rate=0.2
11	Convolution, channel=128, filter=3x3, relu
12	MaxPooling, kernel=2x2
13	Flatten
14	Dense = 256, relu
15	Dropout, rate=0.2
16	Dense = 10, softmax

layer	modelB
1	Dropout, rate=0.2
2	Convolution, channel=96, filter=3x3, relu
3	Convolution, channel=96, filter=3x3, relu
4	MaxPooling, kernel=3x3, stride=2
5	Dropout, rate=0.5
6	Convolution, channel=192, filter=3x3, relu
7	Convolution, channel=192, filter=3x3, relu
8	MaxPooling, kernel=3x3, stride=2
9	Dropout, rate=0.5
10	Convolution, channel=192, filter=3x3, relu
11	Convolution, channel=192, filter=1x1, relu
12	Convolution, channel=10, filter=1x1, relu
13	GlobalAveragePooling
14	Dense = 10, softmax

model A 是普通的 CNN，由 convolution, maxpooling, fully connected neural network 組成，其中我加上了 dropout 來做 regularization。

model B 是我查到的一種作法，增加 convolution layer 而幾乎沒有 fully connected neural network，我猜想在 data 量少的時候多利用 convolution layer 去抓取圖片特徵搞不好會有較好的表現故嘗試了這種作法。

這兩種 model 同樣使用 adam , batch size=32, epoch=60 去 train，表現差不多。兩者精確度只差 1% 左右，kaggle public data 上的 accuracy 為 0.55 上下。

2.Semi-supervised learning (1)

第一個方法我使用 self-training + ImageDataGenerator。

我首先利用 ImageDataGenerator 將 labeled data 做翻轉、伸縮等等產生更多的 data 來 train 先前在 supervised learning 提到的 CNN model A，大概要 train 130 epoch。之後用該 model A 去預測 unlabeled data。如果 model A 預測該 unlabeled data 屬於某個 class 的機率大於 0.9 (我試過是 0.9 表現比較好，太高，太低都不好) 則認為該 unlabeled data 屬於某個 class 的可信度夠高，直接就當它是那個 class，如此大概可以從 45000 筆 unlabeled data 中切出 17000 筆當作新的 data。

接下來我用全新得到的 data 和舊 data 去 train 一個新的 model A。 使用這些新 data 時同樣有用 ImageDataGenerator 來產生更多 data，epoch = 30

多使用 self-training 在 kaggle 上可以達到 0.69 的準確度，相較於只使用 Model A + ImageDataGenerator 的 0.65 準確度有明顯的提高。

3.Semi-supervised learning (2)

在第二個方法中，我使用 autoencoder + ImageDataGenerator。

首先我先使用了全部的 image data 總共 5000(labeled) + 45000(unlabeled) = 50000 筆來 train 我的 autoencoder (training 時似乎會因為初始值的問

題有時候 loss 在開始時就卡住而不會下降，這時候重來多試幾次就可以了)。 之後只用 labeled data，利用 autoencoder 的 encode 部分來取 feature 去 train 一個 fully connected neural network。 model 的詳細在下圖

```
layer                   autoencoder
1       Convolution, channel=96, filter=3x3, relu
2       MaxPooling, kernel=2x2
3       Convolution, channel=144, filter=3x3, relu
4       MaxPooling, kernel=2x2
5       Convolution, channel=192, filter=3x3, relu
6       MaxPooling, kernel=2x2
7       Convolution, channel=192, filter=3x3, relu
8       UpSampling, kernel=2x2
9       Convolution, channel=144, filter=3x3, relu
10      UpSampling, kernel=2x2
11      Convolution, channel=96, filter=3x3, relu
12      UpSampling, kernel=2x2
13      Convolution, channel=3, filter=3x3, relu
layer1~6 = encoder

layer                   encoder + NN
1       Convolution, channel=96, filter=3x3, relu
2       MaxPooling, kernel=2x2
3       Convolution, channel=144, filter=3x3, relu
4       MaxPooling, kernel=2x2
5       Dropout, rate=0.2
6       Convolution, channel=192, filter=3x3, relu
7       MaxPooling, kernel=2x2
8       Dropout, rate=0.2
9       Flatten
10      Dense = 512, relu
11      Dropout, rate=0.5
12      Dense = 10, softmax
```

在這裡我還嘗試了兩種作法，一種是 train NN 時 encoder 部分的參數完全沒有變。一種是在 train NN 時跟著調整 encoder 的最後兩層 convolution layer 的參數(以上圖來說是 encoder+NN 的 layer 3, layer6)，也就是所謂的 fine-tuning。

在 training 時仍是使用 ImageDataGenerator 增加 data 量。

作法 1 在 training 時即能感覺到不是很成功，loss 還很大的時候就下降的非常緩慢，猜想是因為我的 autoencoder 太簡單所以抓出來的 feature 代表性不夠，只用這些 feature 不足以 train 出一個好的 model。

作法 2 在 validation 與 kaggle 上的表現還不錯，經過 50 epoch 在 500 筆 validation data 上的準確度是 0.66 上下, kaggle public data 上的表現是 0.68，相對於純粹 CNN 有蠻大的進步。

4. Result analysis

在這邊我總結各個 model 的成績，因為其實各 model 在 validation data 上的表現與在 kaggle public data 上的表現是差不多的，所以我在這邊只用 kaggle public score 來做比較。 batch_size 全部都是使用 32， optimizer 為 adam

1. CNN model A(B), epoch:60, acc: 0.55
2. CNN model A + self-training, epoch:30, acc:0.55
3. CNN model A + ImageDataGenerator, epoch:130, acc: 0.65
4. CNN model A + ImageDataGenerator + self-training, epoch: 130 + 30, acc:0.69
5. Autoencoder + fine-tuning, epoch:30 (train autoencoder) + 100 (fine-tuning), acc:0.68

能發現使用簡單的 self-training 或 autoencoder 將 unlabeled data 一併考慮進去便可以提高 model 的準確度 4% 左右。而改善 model 表現最大的是 ImageDataGenerator，如果沒有它直接 self-training 我們幾乎不會得到什麼改善，猜想是因為 model 準確度太低而有太多分類錯誤的 data，所以新的 data 沒有什麼用，故 self-training 的效果有蠻大部分是決定於 predict pseudo label 的 model 的準確度。 如果運算資源足夠我想可以嘗試先架一個夠深的 autoencoder 來 train 個準確度高的 model 後再做 self-training，或許可以有更進一步的改善。