Machine Learning HW#3 Report

Semi-supervised Learning 電機四 b02901126 張大方

整份作業使用以 Tensorflow 為 backend 的 keras，image_dim_ordering='th'

1. **Supervised Learning (image generator)**

```
model.add(Convolution2D(64, 3, 3, border_mode='same'))
model.add(Activation('relu'))    #3
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))    #4
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

在我 CNN 的 model 中，總共有三層如左圖所示的結構，最後經由三層 Dense (512) (128) (10)，得到 softmax 後的結果。

```
model.compile(loss='categorical_crossentropy', optimizer=ADAM, metrics=['accuracy'])
```

Training data	accuracy
Labeled data	0.467
Data generator	0.641

在這個階段，我嘗試了兩種方式: 單純的 label data 和使用 data generator 產生 data。最後我以 data generator 為基礎做接下來的題目。

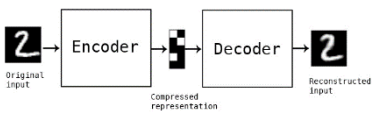
2. **Semi-supervised Learning (self-learning)**

我利用由上一題做出的模型，來標記所有 unlabeled data，**取其中較有信心的 data (probability > 0.95)**加入 labeled data 中，拿來 train 原先的 model。接下來**再用 labeled data 重新 train 同一個 model**，主要是一種類似 **tuning** 的方式，萬一對 unlabeled data 標記的不準確，還可以用這個方法調整回來。我的 self-training process 就是重複上述的過程，經過 6 個 iteration 後停止，得到最終的結果。除了上述的主要過程，我在這個階段對 model 做一一些不同的嘗試，讓結果能變得更好。

1) 加入 **testing data** 到 unlabeled data 中: 增加 data set 大小來提升準確度
2) 在 model 中加入 **batch normalization**: 可防止 "梯度擴散" 。在 BN 中，是通過將 activation 規範為平均值和平方差一致的手段使得原本會減小的 activation 的 scale 變大。是一種更有效的 local response normalization 方法[1]
3) 在 training 過程中，加入 **callbacks**: 可以把最佳的那次 epoch 存下來
4) 更換 activation function, **relu ➔elu**: 不會歸零 input 小於零的值

origin	Add test data	BN	callbacks	Elu
0.71920	0.73180	0.76240	0.77660	0.80300

3. **Semi-supervised Learning (autoencoder self-training)**



利用 autoencoder 將 input data 壓縮，產生一個維度較小的 vector 來代表原本的圖片。在以 DNN 的方法來 train feature。接下來重複 self-training 的步驟，標記 un-labeled data 後選擇其中機率大於 0.95 的，加入原

先的 data set 中。

Autoencoder 的架構參考網站上的 example[2]，encode、decode 皆由**三層 convolution 和 maxpooling** 組成，再將壓縮過的 feature vactor 丟進一個**五層(dense)的 DNN 結構**。而 self-training 的寫法與第二題雷同。

4. Compare and Analyzed the Results

1) 根據第一題 supervised-learning，可以得知在 data 量不足的情況下，**利用 image data generator 可以大幅提高準確度**，這麼做也能幫助接下來的 self-training 過程。因為若是第一階段 labeled data training 不夠準確的話，在標記 unlabeled data 時錯誤會很多，這樣子 self-training 就也沒有什麼幫助，畢竟加入 training data 有可能是錯誤的。

Self-training with labeled data only	Self-training with data generator
0.4670 ➔ 0.50740	0.6410 ➔ 0.70740

2) 在第二題中已有比較加入更多 data 或是不同參數對於準確度的影響，其中 Batch Normalization 的影響很明顯，但是其背後的理論我並沒有完全了解。

3) 在使用 callbacks 時，比較過當要選擇最好的結果時，要**以"val_loss"還是"val_acc"為主**。比較後準確度差異在 10^{-3} 左右，所以我認為並沒有很大的差異。但是總之我還是**選擇了比較高的 val_acc**。

Callbacks on best val_loss	Callbacks on best val_acc
0.76240	0.76320

4) 在我的 model 固定之後，發現將 **batch size 從 128 調整到 64**，竟然使的**準確度從 80.0040 提升到 0.80300**，但是我想這種影響是 case by case 的，剛好適用在這。

5) **Autoencoder 的結果比第二種作法差**太多，只有約 38%的準確度，我想是在於 Autoencoder 沒有使用 data generator，所以由第一題可知，基本上準確度就相差很多了，即便有後來的 self-training，但是一開始爛掉很難救回來了。

5. Reference

[1]Batch Normalization

https://www.zhihu.com/question/38102762

[2]Autoencoder example

https://blog.keras.io/building-autoencoders-in-keras.html

討論同學:

電機四 劉致廷 電機四 陳緯哲