Homework 3 Report -
Image Sentiment Classification

資工四 B04902131 黃郁凱

November 29, 2018

1. 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？
   我主要使用三種類別的 CNN 模型架構，分別是仿 Mobilenet、仿 VGG 和 Kai3。使用原本的 Mobilenet 與 VGG 的成效不好，validation 大約 60%，因此都有做不少的變形。以下是我的模型架構 table1

model	Mobilenet	VGG	Kai3
Convolution Block	conv_bn(1, 16, 1) conv_bn(16, 32, 1) conv_bn(32, 32, 2) conv_dw(32, 64, 1) conv_dw(64, 64, 1) conv_dw(64, 128, 2) conv_dw(128, 256, 2) conv_dw(256, 512, 2)	conv_bn(1,16,1) conv_bn(16,32,1) max_pool conv_bn(32,64,1) conv_bn(64,64,1) max_pool conv_bn(64,128,1) conv_bn(128,128,1) max_pool conv_bn(128,256,1) conv_bn(256,256,1) max_pool	conv_bn(1, 16, 1) conv_bn(16,16,1) conv_bn(16,32,1) max_pool conv_bn(32,64,1) conv_bn(64,64,1) conv_bn(64,128,1) max_pool conv_bn(128,256,1) max_pool conv_bn(128,256,1) max_pool
Linear Block	linear(512*3*3, 512) relu() linear(512, 7)	linear(256*3*3,256) relu() linear(256,32) relu() linear(32, 7)	linear(512*3*3, 512) BatchNorm1d(512) relu() linear(512, 7)

Table 1: 表格中的 conv_bn 是一般 convolution 的基本架構，詳細在 figure 1，而 conv_dw 是 depthwise convolution，也就是 Mobilenet 提出的特殊結構，由兩個 convolution 組成，在 figure 2有詳細內容。表格左方兩者為仿造 Mobilenet 和 VGG16 建造的模型，最終可以得到約 66% 的表現。表格右方為 Kai3 模型的架構，經過前面 Mobilenet 和 VGG 的經驗，我設計出在這個 task 上更適合的模型，前段 filter 數少的部分多一些層數，後半段 filter 數大的層數少，總共四個 max pool，flatten 後再接到 linear 輸出。此模型最終可以達到 69% 的正確率，又比前兩者高了 3%。

$$
\begin{aligned}
conv\_bn(in, out, stride) = Sequetial(& \\
& Conv2d(in, out, 3, stride, 1), \\
& BatchNorm2d(out), \\
& relu())
\end{aligned}
$$

Figure 1: The basic batchnorm convolution block.

$$
\begin{aligned}
conv\_bn(in, out, stride) = Sequetial(& \\
& Conv2d(in, out, 3, stride, 1, groups = in), \\
& BatchNorm2d(out), \\
& relu(), \\
& Conv2d(in, out, 1, 1, 0), \\
& BatchNorm2d(out), \\
& relu())
\end{aligned}
$$

Figure 2: The basic depth-wise convolution block.

Mobilenet 和 VGG16 在 Convolutional Block 的部分和我調整過後有些微差異，他們在深度上更深，並在前半的 filter 數量較少，後半較多；我調整後將深度減少，並在前半段 filter 少的部分多做幾層，後半段大的 filter 層數減少。經過如此調整，正確率可以從原本的 60% 進步到 66%。

由於 Mobilenet 和 VGG 都是用來訓練 Imagenet 專用的模型架構，Imagenet 有一百多萬張相片，分類目標有 1000 個 class；相比我們的 task 規模較小，兩萬多張相片分類 7 個 class。人臉表情辨識的模型不需要如他們一樣龐大的模型，精簡的就可以達到很好的效果。因此經過前人模型的經驗，我自己研發了一套 Kai3 模型可以達到 69% 準確率。

訓練過程使用 batch size 128，learning rate 0.0005。兩者參數均調整過，batch size 過大過小都效果不好；而 learning rate=0.0005 的準確率表現比 0.001 來得好，而更低的 learning rate 又會訓練過慢，所以我選用 0.0005。

我額外將 Kai3 模型做了細微變形，有 Kai, Kai2, Kai4 模型。最終準確率如 table2。

2. 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model，其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

在 table2中可以看出，DNN 架構明顯表現很差，大約 40% 正確率而已，模型架構為多層的 fully connected linear layers，activation function 一樣是 ReLU，訓練 hyperparameters 如同 CNN 的所有模型。因為 DNN 比較難區別小區塊內的特徵，不像 CNN 有 3*3 或 5*5 的 filter，

model name	validation	testing
DNN	0.400	0.416
Mobilenet	0.666	0.660
VGG	0.662	0.667
Kai	0.655	0.675
Kai2	0.668	0.671
Kai3	0.685	0.690
Kai4	0.670	0.681
ensemble	-	**0.702**

Table 2: 不同模型的正確率，validation 切 9:1，testing 為 kaggle 分數。

當遇到相似圖形特徵就會被激發，這就是 CNN 模型會勝過 DNN 的直觀解釋。另外，CNN 使用參數的效率比 DNN 來得好，DNN 在每個 pixel 上都要有一個相對應的參數，而 CNN 則是同一塊 filter 重複利用，會使得 DNN 模型較厚重，參數使用效率不佳，在差不多參數的情況下，表現自然會比較糟。

3. 觀察答錯的圖片中，哪些 class 彼此間容易用混？並說明你觀察到了什麼？[繪出 confusion matrix 分析]
   我用正確率最高的模型 Kai3，切 train:validation = 9:1，train 完的模型在 validation 上的表現畫成 confusion matrix 如圖 figure 3。

   最明顯的是 disgust 這個 class 完全分類不出來，原因是 training data 的比例有明顯的少，所佔的比例只有 1.5%，因為每筆資料答錯被懲罰所占的比重都一樣，所以模型選擇直接放棄判斷這類型的 class，要改善此問題有兩個方法，第一是增加這部分的 training data，使他的比重和其他種類的差不多，第二是增加犯錯懲罰的比重，加在 loss 當中，讓模型知道要特別注重這個 class。

   另外可以發現，沿著斜對角線的格子會對稱，同時錯很多或同時錯很少。例如 sad 和 neutral、fear 和 sad、surprise 和 fear，都是同會有將近 10% 的比例會混淆，而 happy 和 angry、sad 和 surprise 就幾乎不會混淆。這個結果很合乎常理，因為快樂和生氣或是悲傷和驚訝都是反差很大的表情，機器應該可以輕易分別這個不同；相反的，傷心害怕都是臉部表情比較不激烈的肢體語言，機器要分辨就會難度變高許多，因此混淆率增加是合理的。

4. CNN time/space complexity: For a. b. Given a CNN model as

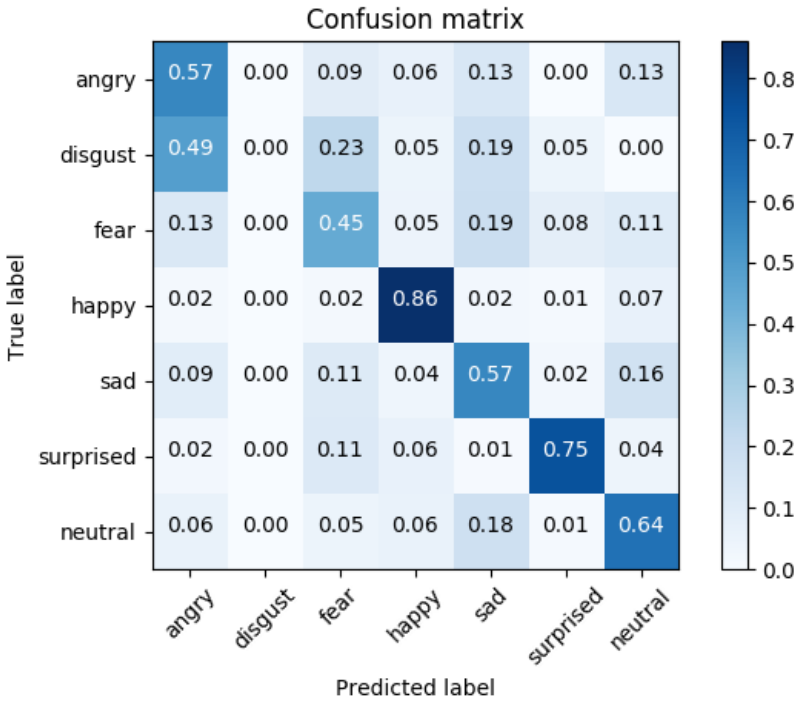

Figure 3: 圖是 normalize 過後的比例，表示 Kai3 模型在 validation set 上的表現，y 軸是正確的 label，x 軸是模型的預測，顏色越深代表比例越重。

```
model = Sequential()
model.add(Conv2D(filters=6,
                 strides=(3, 3),
"""Layer A"""    padding ="valid",
                 kernel_size=(2,2),
                 input_shape=(8,8,5),
                 activation='relu'))
model.add(Conv2D(filters=4,
                 strides=(2, 2),
"""Layer B"""    padding ="valid",
                 kernel_size=(2,2),
                 activation='relu'))
```

And for the c. given the parameter as: kernel size = (k,k); channel size = c; input shape of each layer = (n,n); padding = p; strides = (s,s);

(a) How many parameters are there in each layer?
如果不考慮 bias 的話，考慮的話要再加上 filter 數量的參數。

i. Layer A: 2*2*5*6 = 120

ii. Layer B: 2*2*6*4 = 96

(b) How many multiplications/additions are needed for a forward pass(each layer).(hint: 不用考慮 bias)

i. Layer A:
multiply: 3*3*(2*2*5) * 6 = 1080
addition: 3*3*(2*2*5-1) * 6 = 1026

ii. Layer B:
multiply: 1*1*(2*2*6) * 4 = 96
addition: 1*1*(2*2*6-1) * 4 = 92

(c) What is the time complexity of convolutional neural networks?

- For one layer:
  output size: $(\lfloor \frac{n-k+2p}{s} \rfloor + 1)^2 \cdot c_{out} \leq (\frac{n-k+2p}{s} + 1)^2 \cdot c_{out}$
  total multiplication and addition = $output\_size \cdot (k^2 + k^2 - 1) \cdot c_{in} \leq (\frac{n-k+2p}{s} + 1)^2 \cdot c_{out} \cdot 2(k^2) \cdot c_{in}$
  time complexity is $O((\frac{n-k+2p}{s})^2 c_{in} c_{out} k^2)$.
- Total Layers:
  time complexite is $O(\sum_{i=2}^{l} (\frac{n_i - k_i + 2p_i}{s_i})^2 c_{i-1} c_i k_i^2)$.

5. PCA practice:Problem statement: Given 10 samples in 3D space.
(1,2,3), (4,8,5), (3,12,9), (1,8,5), (5,14,2), (7,4,1), (9,8,9), (3,8,1), (11,5,6), (10,11,7)

(a) What are the principal axes?
Calculate step by step:

i. input

$$X = array([[1,2,3],\\ [4,8,5],\\ [3,12,9],\\ [1,8,5],\\ [5,14,2],\\ [7,4,1],\\ [9,8,9],\\ [3,8,1],\\ [11,5,6],\\ [10,11,7]])$$

ii. covariance matrix

$$Cov = array([[1.486, 0.061, 0.404],\\ [0.061, 1.506, 0.358],\\ [0.404, 0.358, 1.007]])$$

iii. eigen value and eigen vector

$$eigen\_value = array([0.675, 1.435, 1.888])$$
$$eigen\_vector = array([[0.399, -0.678, -0.616],\\ [0.337, 0.734, -0.588],\\ [-0.852, -0.027, -0.522]])$$

iv. principle axis 1: the axis with the largest eigen value 1.888
$[-0.616, -0.588, -0.522]$
principle axis 2: the axis with the second largest eigen value 1.435
$[-0.678, 0.734, -0.027]$
principle axis 3: the last axis with eigen value 0.675
$[0.399, 0.337, -0.852]$

(b) Compute the principal components for each sample.

principle component is the eigen vector sorted with eigen value: $\begin{bmatrix} -0.616 & -0.588 & -0.522 \\ 0.678 & -0.734 & 0.027 \\ 0.399 & 0.337 & -0.852 \end{bmatrix}$

(c) Reconstruction error if reduced to 2D.(Calculate the L2-norm)

$$
\begin{aligned}
Loss &= \frac{min(eigen\_values)}{\sqrt{\sum_{i=1}^{3} eigen\_value_i^2}} \\
&= \frac{0.675}{\sqrt{0.675^2 + 1.435^2 + 1.888^2}} \\
&= 0.273
\end{aligned}
$$