

Homework3 Report

Professor Pei-Yuan Wu
EE5184 - Machine Learning

姓名：歐政鷹

學號：R07922142

1. (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？

Using TensorFlow backend.					
Layer (type)	Output Shape	Param #			
conv2d_1 (Conv2D)	(None, 48, 48, 64)	640	batch_normalization_9 (Batch Normalization)	(None, 6, 6, 512)	2048
batch_normalization_1 (Batch Normalization)	(None, 48, 48, 64)	256	dropout_9 (Dropout)	(None, 6, 6, 512)	0
dropout_1 (Dropout)	(None, 48, 48, 64)	0	conv2d_10 (Conv2D)	(None, 6, 6, 512)	2359808
conv2d_2 (Conv2D)	(None, 48, 48, 64)	36928	batch_normalization_10 (Batch Normalization)	(None, 6, 6, 512)	2048
batch_normalization_2 (Batch Normalization)	(None, 48, 48, 64)	256	max_pooling2d_4 (MaxPooling2D)	(None, 3, 3, 512)	0
max_pooling2d_1 (MaxPooling2D)	(None, 24, 24, 64)	0	dropout_10 (Dropout)	(None, 3, 3, 512)	0
dropout_2 (Dropout)	(None, 24, 24, 64)	0	conv2d_11 (Conv2D)	(None, 3, 3, 512)	2359808
conv2d_3 (Conv2D)	(None, 24, 24, 128)	73856	batch_normalization_11 (Batch Normalization)	(None, 3, 3, 512)	2048
batch_normalization_3 (Batch Normalization)	(None, 24, 24, 128)	512	dropout_11 (Dropout)	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 24, 24, 128)	0	conv2d_12 (Conv2D)	(None, 3, 3, 512)	2359808
conv2d_4 (Conv2D)	(None, 24, 24, 128)	147584	batch_normalization_12 (Batch Normalization)	(None, 3, 3, 512)	2048
batch_normalization_4 (Batch Normalization)	(None, 24, 24, 128)	512	dropout_12 (Dropout)	(None, 3, 3, 512)	0
max_pooling2d_2 (MaxPooling2D)	(None, 12, 12, 128)	0	conv2d_13 (Conv2D)	(None, 3, 3, 512)	2359808
dropout_4 (Dropout)	(None, 12, 12, 128)	0	batch_normalization_13 (Batch Normalization)	(None, 3, 3, 512)	2048
conv2d_5 (Conv2D)	(None, 12, 12, 256)	295168	max_pooling2d_5 (MaxPooling2D)	(None, 1, 1, 512)	0
batch_normalization_5 (Batch Normalization)	(None, 12, 12, 256)	1024	dropout_13 (Dropout)	(None, 1, 1, 512)	0
dropout_5 (Dropout)	(None, 12, 12, 256)	0	flatten_1 (Flatten)	(None, 512)	0
conv2d_6 (Conv2D)	(None, 12, 12, 256)	590080	dense_1 (Dense)	(None, 256)	131328
batch_normalization_6 (Batch Normalization)	(None, 12, 12, 256)	1024	batch_normalization_14 (Batch Normalization)	(None, 256)	1024
dropout_6 (Dropout)	(None, 12, 12, 256)	0	dropout_14 (Dropout)	(None, 256)	0
conv2d_7 (Conv2D)	(None, 12, 12, 256)	590080	dense_2 (Dense)	(None, 256)	65792
batch_normalization_7 (Batch Normalization)	(None, 12, 12, 256)	1024	batch_normalization_15 (Batch Normalization)	(None, 256)	1024
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	0	dropout_15 (Dropout)	(None, 256)	0
dropout_7 (Dropout)	(None, 6, 6, 256)	0	dense_3 (Dense)	(None, 64)	16448
conv2d_8 (Conv2D)	(None, 6, 6, 512)	1180160	batch_normalization_16 (Batch Normalization)	(None, 64)	256
batch_normalization_8 (Batch Normalization)	(None, 6, 6, 512)	2048	dropout_16 (Dropout)	(None, 64)	0
dropout_8 (Dropout)	(None, 6, 6, 512)	0	dense_4 (Dense)	(None, 7)	455
conv2d_9 (Conv2D)	(None, 6, 6, 512)	2359808	Total params: 14,946,759		
			Trainable params: 14,937,159		
			Non-trainable params: 9,600		
			None		

以上是我的 **CNN** 模型架構，主要是參考了 **VGG-16** 的架構，並加入漸進式的 **dropout layer**、**batch normalization layer** 及對某幾層的 **filters** 數進行了調整而得出。

在訓練開始前，我取了 10% 的資料作為 validation data，其餘的資料則利用了 image data augmentation 的技巧對圖片進行了旋轉、平移、縮放等不同方式的前處理，增加訓練資料的數量。接著再利用我的模型架構在 batch_size=128, epochs=300 的情況下進行訓練。在每層 layer，我都是以 padding='same', kernel_initializer='glorot_normal', activation='relu' 作為基本的設定。

在訓練過程中，我利用 keras 的 ModelCheckpoint 來幫助我找出所有 epochs 中 validation test 中結果最好的 model。最後分別取得了 Public Score = 0.65422 及 Private Score = 0.65923 的準確度。

2. (1%) 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？

Layer (type)	Output Shape	Param #		
=====				
flatten_1 (Flatten)	(None, 2304)	0	batch_normalization_6 (Batch Normalization)	4096
dense_1 (Dense)	(None, 256)	590080	dropout_6 (Dropout)	0
batch_normalization_1 (Batch Normalization)		1024	dense_7 (Dense)	1049600
dropout_1 (Dropout)	(None, 256)	0	batch_normalization_7 (Batch Normalization)	4096
dense_2 (Dense)	(None, 256)	65792	dropout_7 (Dropout)	0
batch_normalization_2 (Batch Normalization)		1024	dense_8 (Dense)	2099200
dropout_2 (Dropout)	(None, 256)	0	batch_normalization_8 (Batch Normalization)	8192
dense_3 (Dense)	(None, 512)	131584	dropout_8 (Dropout)	0
batch_normalization_3 (Batch Normalization)		2048	dense_9 (Dense)	4196352
dropout_3 (Dropout)	(None, 512)	0	batch_normalization_9 (Batch Normalization)	8192
dense_4 (Dense)	(None, 512)	262656	dropout_9 (Dropout)	0
batch_normalization_4 (Batch Normalization)		2048	dense_10 (Dense)	4196352
dropout_4 (Dropout)	(None, 512)	0	batch_normalization_10 (Batch Normalization)	8192
dense_5 (Dense)	(None, 1024)	525312	dropout_10 (Dropout)	0
batch_normalization_5 (Batch Normalization)		4096	dense_11 (Dense)	14343
dropout_5 (Dropout)	(None, 1024)	0	=====	
dense_6 (Dense)	(None, 1024)	1049600	Total params: 14,223,879	
			Trainable params: 14,202,375	
			Non-trainable params: 21,504	

以上是我的 DNN 模型架構，跟 CNN 架構一樣擁有漸進式的 dropout layer 及 batch normalization layer。

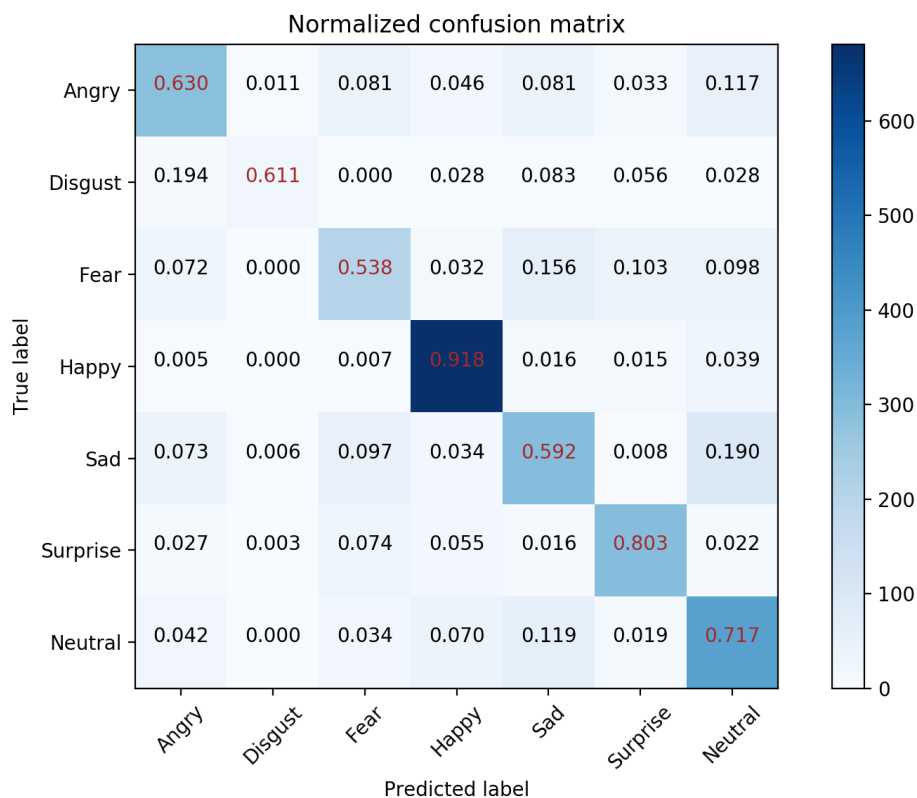
訓練過程與 CNN 大同小異，10% 的資料作為 validation data，其餘的資料則利用了 image data augmentation 的技巧對圖片進行了旋轉、平移、縮放等不同方式的前處理，增加訓練資料的數量。

模型架構在 batch_size=128, epochs=300 的情況下進行訓練。在每層 layer，我都是以 padding='same', activation='relu' 作為基本的設定。

在訓練過程中，也利用 keras 的 ModelCheckpoint 來幫助我找出所有 epochs 中 validation test 中結果最好的 model。最後分別取得了 Public Score = 0.30844 及 Private Score = 0.32153 的準確度。

與 CNN 比較，根據我的 model 檔案紀錄，我發現 DNN 會在比較短時間內出現收斂的情況。其次 DNN 的運算速度比 CNN 快很多，這是因為在卷積過程中會有很多的運算。最後很明顯的 DNN 的準確度遠不及 CNN。

3. (1%) 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]



很明顯的看的出來，disgust 比較容易誤判成 angry、sad 容易誤判成 neutral、fear 跟 neutral 容易誤判成 sad。

我有去 data set 中找出容易被誤判的幾類照片來觀察，發現有些照片比較起來看下去第一瞬間真的不好判斷說應該要分到哪一類，所以程式分不出來我覺得也是正常的。

-----Handwritten question-----

4. (1.5%, each 0.5%) CNN time/space complexity:
For a. b. Given a CNN model as

```

model = Sequential()
model.add(Conv2D(filters=6,
                  strides=(3, 3),
                  padding="valid",
                  kernel_size=(2,2),
                  input_shape=(8,8,5),
                  activation='relu'))
model.add(Conv2D(filters=4,
                  strides=(2, 2),
                  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(Hint: you may consider whether the number of parameter is related with)

Layer A:

$$2*2*6*5 + 6 = 126$$

Layer B:

$$2*2*4*6 + 4 = 100$$

- b. How many multiplications/additions are needed for a forward pass(each layer).

Layer A:

$$\text{multiplications : } 2*2*5*3*3*6 = 1080$$

$$\text{additions : } (2*2*5 - 1)*3*3*6 = 1026$$

Layer B:

$$\text{multiplications : } 2*2*6*1*1*4 = 96$$

$$\text{additions : } (2*2*6 - 1)*1*1*4 = 92$$

- c. What is the time complexity of convolutional neural networks?(note: you must use big-O upper bound, and there are l(lower case of L) layer, you can use

$\square, \square, \square_{-1}$ as lth and l-1th layer)

5. (1.5%,each 0.5%)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. (1) What are the principal axes?

- b. (2) Compute the principal components for each sample.

(7.18658682, 1.37323947),
(0.75871342, -0.94399334),
(-3.07034019, -4.45059025),
(2.60849751, -2.97853006),
(-1.82299166, -4.75401212),
(3.35457763, 3.91896138),
(-4.41464321, 2.55604371),
(3.46569126, -1.73131477),
(-2.31359638, 6.03371503),
(-5.75249521, 0.97648096)

c. (3) Reconstruction error if reduced to 2D.(Calculate the L2-norm)