Introduction to Digital Image Processing

Homework 3

0510002 袁鈺勛

A. 基礎修改

1. Activation function

a. Tanh

保持不變原本每個 model 的 activation

```
model.add(Activation('tanh'))
```

得到的 accuracy 結果如下

```
Over Acu: 0.7678309858475615
```

b. ReLU

將每個 model 的 activation 改成'relu'

```
model.add(Activation('relu'))
```

得到的 accuracy 結果如下

```
Over Acu: 0.23116197569875174
```

c. Sigmoid

將每個 model 的 activation 改成'sigmoid'

```
model.add(Activation('sigmoid'))
```

得到的 accuracy 結果如下

```
Over Acu: 0.76389863695187703
```

tanh	ReLU	Sigmoid
0.768	0.231	0.764

以以上三個結果，所以後面採用原先的 tanh

2. 深度

a. 3 層

不更改原先的 code

```
# First Layer
model.add(Conv1D(nb_filter=5,
                 filter_length=10,
                 init='glorot_uniform',
                 border_mode='same',
                 input_shape=(train_params['max_size'], 3),
                 bias=True))

model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of First Layer

# Second Layer
model.add(Conv1D(nb_filter=10,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))

model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Second Layer

# Third Layer
model.add(Conv1D(nb_filter=20,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))
model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Third Layer
```

得到的結果如下

```
Over Acu: 0.7665604517718305
```

b. 4 層

將網路結構改成 4 層

```
# First Layer
model.add(Conv1D(nb_filter=5,
                 filter_length=10,
                 init='glorot_uniform',
                 border_mode='same',
                 input_shape=(train_params['max_size'], 3),
                 bias=True))

model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of First Layer

# Second Layer
model.add(Conv1D(nb_filter=10,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))

model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Second Layer

# Third Layer
model.add(Conv1D(nb_filter=20,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))
model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Third Layer

# Fourth Layer
model.add(Conv1D(nb_filter=20,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))
model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Fourth Layer
```

得到的 accuracy 結果如下

```
Over Acu: 0.7668933192238192
```

c. 5 層

將網路結構改成 5 層

```
# First Layer
model.add(Conv1D(nb_filter=5,
                 filter_length=10,
                 init='glorot_uniform',
                 border_mode='same',
                 input_shape=(train_params['max_size'], 3),
                 bias=True))

model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of First Layer

# Second Layer
model.add(Conv1D(nb_filter=10,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))

model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Second Layer

# Third Layer
model.add(Conv1D(nb_filter=20,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))
model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Third Layer

# Fourth Layer
model.add(Conv1D(nb_filter=20,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))
model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Fourth Layer

# Fifth Layer
model.add(Conv1D(nb_filter=20,
                 filter_length=20,
                 init='glorot_uniform',
                 border_mode='same',
                 bias=True))
model.add(Activation('tanh'))

model.add(MaxPooling1D(pool_size=2))
# End of Fifth Layer
```

得到的 accuracy 結果如下

```
Over Acu: 0.7659119247675089
```

3 層	4 層	5 層
0.7666	0.7669	0.7659

以以上三個結果，後面採用 4 層的網路結構

3. Pooling function
   a. Max pooling

      不更動原先的 pooling function

```
model.add(MaxPooling1D(pool_size=2))
```

得到的 accuracy 結果如下

```
Over Acu: 0.7641088236323329
```

b. Average pooling

將每個 model 的 pooling function 改成 AveragePooling1D

```
model.add(AveragePooling1D(pool_size=2))
```

得到的 accuracy 結果如下

```
Over Acu: 0.7657828206937505
```

Max pooling	Average pooling
0.764	0.766

以以上兩個結果，後面採用 average pooling

B. 網路架構

1. 架構

Type	Size/Stride	Number of channels
Input	M	3
Conv1D, tanh	10 x 1	5
AveragePool	2 x 1/2 x 1	-
Conv1D, tanh	20 x 1	10
AveragePool	2 x 1/2 x 1	-
Conv1D, tanh	20 x 1	20
AveragePool	2 x 1/2 x 1	-
Conv1D, tanh	20 x 1	20
AveragePool	2 x 1/2 x 1	-
Global-avg-pool	-	-
Fully-con, tanh	20	-
Dropout (0.3)	-	-
Fully-con, Softmax	2	-

2. Epoch

a. 5 epochs

更改 epoch 數目為 5

```
train_params = {'batch_size':256,
        'max_size':256,
        'base_lr':0.001,
        'decay_steps':5,
        'decay_factor':0.5,
        'num_epochs':5,
        'neg_samples':len(data[0]),
        'pos_samples':len(data[1]),
        'total_samples':len(data[0])+len(data[1]),
        'checkpoint':1}
```

更改 test 所使用的 h5 檔案

```
model = load_model('model_4.h5')
```

得到的 accuracy 結果如下

```
Over Acu: 0.7657828206937505
```

b. 10 epochs

更改 epoch 數目為 10

```
train_params = {'batch_size':256,
        'max_size':256,
        'base_lr':0.001,
        'decay_steps':5,
        'decay_factor':0.5,
        'num_epochs':10,
        'neg_samples':len(data[0]),
        'pos_samples':len(data[1]),
        'total_samples':len(data[0])+len(data[1]),
        'checkpoint':1}
```

更改 test 所使用的 h5 檔案

```
model = load_model('model_9.h5')
```

得到的 accuracy 結果如下

```
Over Acu: 0.7657828206937505
```

c. 15 epochs

更改 epoch 數目為 15

```
train_params = {'batch_size':256,
        'max_size':256,
        'base_lr':0.001,
        'decay_steps':5,
        'decay_factor':0.5,
        'num_epochs':15,
        'neg_samples':len(data[0]),
        'pos_samples':len(data[1]),
        'total_samples':len(data[0])+len(data[1]),
        'checkpoint':1}
```

更改 test 所使用的 h5 檔案

```
model = load_model('model_14.h5')
```

得到的 accuracy 結果如下

```
Over Acu: 0.7686456470284244
```

d. 20 epochs

更改 epoch 數目為 20

```
train_params = {'batch_size':256,
        'max_size':256,
        'base_lr':0.001,
        'decay_steps':5,
        'decay_factor':0.5,
        'num_epochs':20,
        'neg_samples':len(data[0]),
        'pos_samples':len(data[1]),
        'total_samples':len(data[0])+len(data[1]),
        'checkpoint':1}
```

更改 test 所使用的 h5 檔案

```
model = load_model('model_19.h5')
```

得到的 accuracy 結果如下

```
Over Acu: 0.7673463301609267
```

5 epoch	10 epoch	15 epoch	20 epoch
0.7658	0.7658	0.7686	0.7673

以以上四個結果，accuracy 會隨著 epoch 上升而上升

C. 心得

這次調整各種參數進行訓練，訓練出來的結果有些都有點奇怪，像是調整不同的 activation function，被廣泛使用且效果較好的 ReLU 卻是最差的結果，而

tanh 和 sigmoid 基本上是差不多的圖形，所以結果差不多不太意外。還有在調整層數方面，資料量很大，應該還不至於 overfitting，可是 accuracy 卻沒有呈現上升趨勢，4 層比 5 層還優秀。在 pooling 方面，針對邊緣偵測，應該是 max pooling 的效果會比 average pooling 還要好，但得到的結果卻是 average pooling 比較好，感覺很奇怪。完成網路架構後，要測試不同 epoch 數目所訓練出來的網路 accuracy，從結果來看還算正常，隨著 epoch 數量上升，理論上應該要學習的更精確。在一開始做這個作業並完成第一次的訓練後，我想用 test 測試 accuracy 如何，結果卻跳出了 true_divide 遇到了 invalid value，而且 np.where(pred == pred.max())[1][0]也遇到了 index error，沒有 0 這個 index，接下來我開始一步一步確認問題，首先發現了 np.where(pred == pred.max())[1]是空的 list，再來又確認到了 np.where(pred == pred.max())是一個有兩個空 list 的 list，所以知道了在 true_divide 的 invalid value 一定有大問題，然後用 np.count_nonzero 確認到 given_rep 裡全都是 0，這才發現 true_divide 是因為用 0 來除其他 element，難怪會出問題。