csc420a5

Guanchen Zhang

November 2018

1 Question 1

1.1

It is not a good idea to train a neural network to classify an input image into either one of the employees and an unknown category because training set is too small for training a neural network for each face (5 images). It is likely that the features of each person are not well-captured so the query image and images in database have a low similarity in all cases, making the identification of unknown person even harder.

1.2

The term frequency weights words occuring often in a document (local importance) because if a word appears a lot of times, then the word must be important. The inverse document frequency indicates that we are able to downweight the words that occur often in the full dataset because for a word to be considered a signature word of a document, it shouldn't appear that often in the other documents. Thus, for a word to have high tf-idf in a document, it must appear a lot of times in said document and must be absent in the other documents. It must be a signature word of the document. When computing the similarity, we focus on the similarity between the signature word of reference and signature word of dataset. Therefore, the tf-idf works better than just present/absent representation

1.3

Clustering help us divide embedding (descriptor) into different groups. Since similar embeddings are close to each other in distance and in our case, each training image is turned into an embedding, ideally one cluster should only contain the images of a specific person. Therefore, when we do a query, it is not necessary to look over the whole database for matches, instead, we can just compare the query image with the cluster center which is most representative (mean) in that cluster.

1.4

The visual words are "representative" vectors in that cluster, which can be the cluster center.

1.5

Using K-means clustering, I cluster saved embedding into 6 cluster which is the number of person. With the cluster center as the representation of the cluster, I compute the similarity between embedding of these centers and the embedding of query image using the formula from lecture slides. The visual word with largest similarity is picked and the query image is classified to this visual word. If the similarity is too small (smaller than a threshold), it is highly likely that query image belongs to a unknown person.

1.6

The prediction for each query image is (for simplicity, I didn't write the full file name):

```
face 97 : [9, 10, 12, 13, 26, 34, 39, 42, 43, 53, 59, 63, 64, 65, 68,
72, 73, 75, 85, 91, 94, 97, 102, 103, 107, 111, 113, 132, 136, 142,
170, 178, 183, 191]
face 107 : [4, 6, 15, 19, 22, 38, 47, 50, 58, 69, 87, 95, 96, 109,
130, 131, 133, 138, 143, 144, 158, 160, 175, 179, 188]
face 109 : [32, 33, 45, 54, 57, 62, 77, 80, 84, 89, 98, 100, 105,
106, 108, 117, 119, 122, 135, 140, 145, 157, 161, 165, 180, 186, 187,
189, 192, 194, 197]
face 116 : [8, 17, 18, 20, 27, 28, 37, 40, 44, 55, 60, 67, 74, 76,
82, 86, 90, 92, 93, 99, 110, 118, 121, 126, 127, 128, 129, 134, 137,
141, 147, 151, 153, 155, 159, 163, 167, 173, 181, 185, 196]
face 119 : [9, 10, 12, 13, 26, 34, 39, 42, 43, 53, 59, 63, 64, 65,
68, 72, 73, 75, 85, 91, 94, 97, 102, 103, 107, 111, 113, 132, 136,
142, 170, 178, 183, 191]
face 126 : [3, 21, 24, 25, 29, 30, 35, 41, 46, 49, 51, 71, 78, 104,
114, 115, 120, 123, 124, 125, 139, 146, 150, 152, 164, 166, 168, 169,
171, 174, 176, 182, 193, 195]
face 600 : no matching
```

1.7

2 Question 2

2.1

when we do back-propagation and calculating gradients of loss with respect to the weights , the gradients tends to get smaller and smaller as we keep on moving

backward in the Network because the backward pass is linear. This means that the neurons in the Earlier layers learn very slowly as compared to the neurons in the later layers in the Hierarchy. The Earlier layers in the network are slowest to train.
During neural network training with backpropagation, the (local) minimum of the error function is found by iteratively taking small steps in the direction of the negative error derivative with respect to networks weights (i.e. gradients). With each subsequent layer the magnitude of the gradients gets exponentially smaller (vanishes) thus making the steps also very small which results in very slow learning of the weights in the lower layers of a deep network.

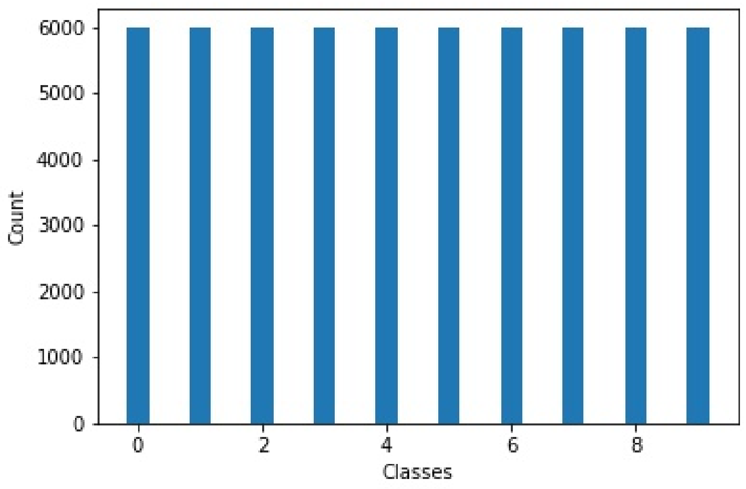
One of the factors causing the gradients to shrink when going through layers are the activation function derivatives (derivative of layers output wrt. its input).
We can avoid this problem by using activation functions which don't have this property of 'squashing' the input space into a small region. A popular choice is Rectified Linear Unit which maps x to max(0,x).

2.2

Max pooling does two main things:

Reduces the number of parameters within the model (as you can observe above, the output is smaller than the input)
It generalizes the results from a convolutional filter - making the detection of features invariant to scale or orientation changes.

2.3



The dataset is split into training set and validation set in 6:1.

2.4

In this part, I used 2 layers fully connected model with ReLu as an activation function after first layer and softmax as an activation function after the second layer. The formula for ReLu is $y_k = max(0, z_k)$ and for softmax is $y_k = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}$ where z_k is the linear prediction

2.5

The categorical loss function is

$$loss = \frac{1}{N}\sum_{i=1}^{N} -t^{(i)}log(y^{(i)}) - (1-t^{(i)})log(1-y^{(i)})$$

where N is number of observations, $y^{(i)}$ is the i-th model prediction and $t^{(i)}$ is the i-th ground truth label.
In matrix form:

$$loss = -\sum_{k=1}^{K} t^k logy^k = -t^T(logy)$$

where K is the output dimension and log is in element-wise.

2.6

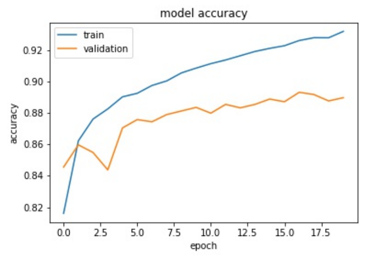


Figure 1: training validation accuracy batch = 64 epoch =20

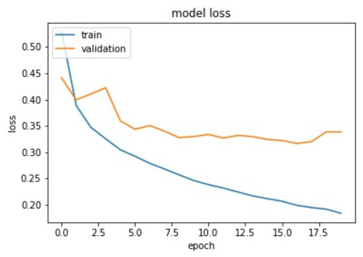


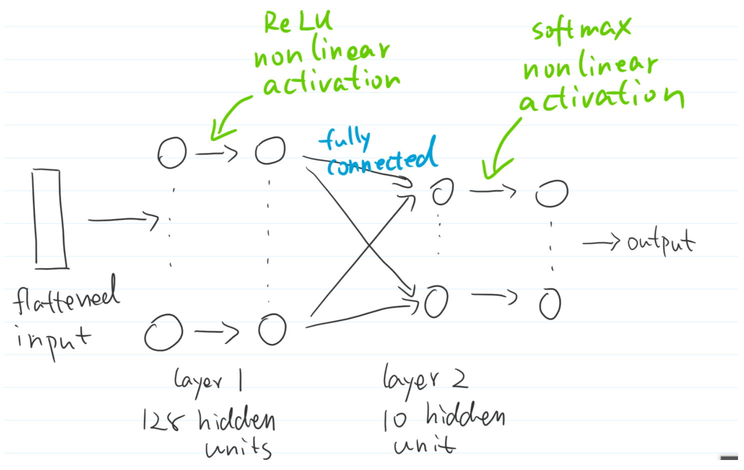
Figure 2: training validation loss batch = 64 epoch =20

The batch size I chose is 64. From the plot we notice that the training accuracy kept increasing and training loss kept decreasing as we do more epoches. But at the same time, the validation loss and accuracy fluctuated, even dropped, indicating overfitting. Therefore, we should stop training before the model loses its generalization on prediction. Therefore, I should in my model, i used epoch = 20, batch size = 64, learning rate = 0.001.

after 20 epoches the final training loss: 0.1842 - acc: 0.9322 - val loss: 0.3386 - val acc: 0.8899

The architecture is:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_17 (Flatten)         (None, 784)               0
_________________________________________________________________
dense_34 (Dense)             (None, 128)               100480
_________________________________________________________________
dense_35 (Dense)             (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
```



2.7

If we have a too small batch size, we cannot exploit vectorization and each epoch takes a long time. If the batch size is too large, it takes more memory to store activations and longer to compute each gradient update. Therefore, it is good to have a batch size that is not too large nor too small. The accuracy and loss indicate that the model with large batch size has a good performance on early stages eg. first few epoches, because the variance is reduced during mini-batch.

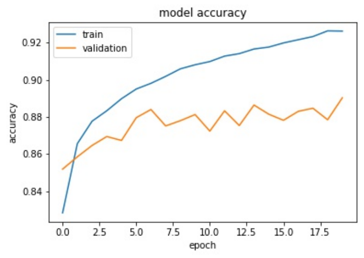


Figure 3: training validation accuracy batch = 8

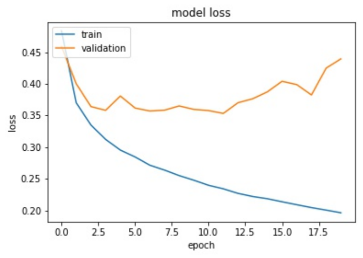


Figure 4: training validation loss batch = 8

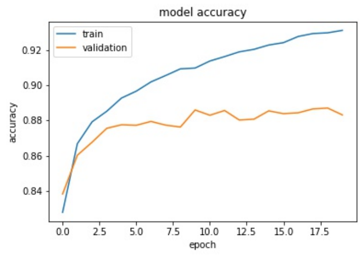


Figure 5: training validation accuracy batch = 16

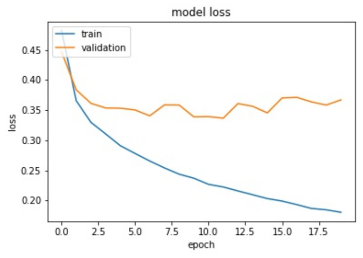


Figure 6: training validation loss batch = 16

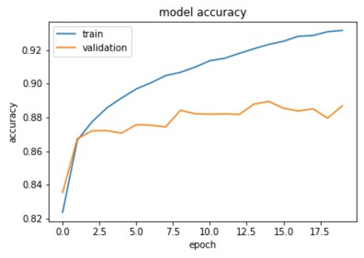


Figure 7: training validation accuracy batch = 32

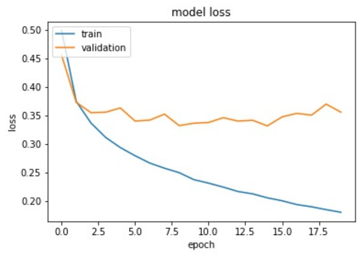


Figure 8: training validation loss batch = 32

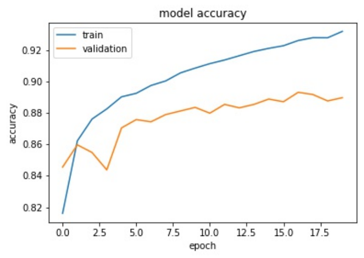


Figure 9: training validation accuracy batch = 64

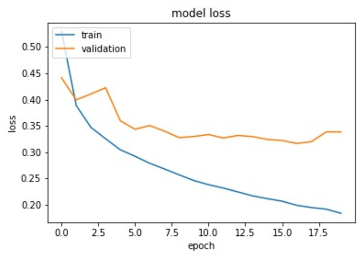


Figure 10: training validation loss batch = 64