CSC420 Assignment5

Name: Yilin Qu

Student number: 1002995734

Utorid: quyilin

Question1

(a)

No it is not a good idea to train a neural net to classify these employees because we have too few images for every person, so the model we trained might be too simple and gives the wrong classification.

(b)

TF-IDF will reweight every entry so that:

- It is easier to extract the most descriptive features in the document
- It weights down the highly frequency term and weights up the more unique terms
- Can easily compute the similarity between two documents using TF-IDF

(c)

(see Q1.py)

(d)

Clustering the embedding (in this case 6) is to do KMeans in the next step to classify all images in "saved_faces" to 6 categories, so that each cluster center can represent different classes of input faces, and also we can compute the similarity faster using these cluster centers.

(e)

(KMeans process in Q1.py)

Representation: The cluster center, the 128-dimention embedding vector.

(f)

(see Q1.py)

(g)

For matching images, I will use the inverted_index dictionary before, as well as the KMeans labels and cluster_centers. The process is as follow:

- Use the same model to compute the embedding for every image in "input_faces"
- Use these embeddings to compute a similarity score among all six cluster

centers. (similarity is dot product over the product of their norms)

- Use a threshold to filter the scores and only keep the higher ones.
- Return the max score after thresholding and its corresponding category.

After some experiments, I found that set threshold to 0.8 will classify out the “wrong image”, i.e., face_image_600.jpg, which has no match in our saved_faces dataset.

(h)

(in Q1.py)

output:

{**'face_image_107.jpg'**: [['face_image_112.jpg', 'face_image_115.jpg', 'face_image_117.jpg', 'face_image_130.jpg', 'face_image_133.jpg', 'face_image_138.jpg', 'face_image_143.jpg', 'face_image_144.jpg', 'face_image_145.jpg', 'face_image_149.jpg', 'face_image_15.jpg', 'face_image_154.jpg', 'face_image_157.jpg', 'face_image_158.jpg', 'face_image_160.jpg', 'face_image_175.jpg', 'face_image_187.jpg', 'face_image_188.jpg', 'face_image_19.jpg', 'face_image_190.jpg', 'face_image_30.jpg', 'face_image_31.jpg', 'face_image_32.jpg', 'face_image_38.jpg', 'face_image_4.jpg', 'face_image_47.jpg', 'face_image_50.jpg', 'face_image_58.jpg', 'face_image_6.jpg', 'face_image_69.jpg', 'face_image_7.jpg', 'face_image_70.jpg', 'face_image_81.jpg', 'face_image_87.jpg', 'face_image_9.jpg', 'face_image_91.jpg', 'face_image_95.jpg', 'face_image_96.jpg'], 3], **'face_image_109.jpg':** [['face_image_104.jpg', 'face_image_109.jpg', 'face_image_114.jpg', 'face_image_120.jpg', 'face_image_123.jpg', 'face_image_124.jpg', 'face_image_125.jpg', 'face_image_131.jpg', 'face_image_139.jpg', 'face_image_140.jpg', 'face_image_146.jpg', 'face_image_150.jpg', 'face_image_152.jpg', 'face_image_164.jpg', 'face_image_166.jpg', 'face_image_168.jpg', 'face_image_169.jpg', 'face_image_171.jpg', 'face_image_174.jpg', 'face_image_176.jpg', 'face_image_179.jpg', 'face_image_182.jpg', 'face_image_189.jpg', 'face_image_193.jpg', 'face_image_21.jpg', 'face_image_22.jpg', 'face_image_24.jpg', 'face_image_25.jpg', 'face_image_29.jpg', 'face_image_3.jpg', 'face_image_35.jpg', 'face_image_41.jpg', 'face_image_51.jpg', 'face_image_71.jpg', 'face_image_78.jpg'], 4],

```
'face_image_116.jpg': [['face_image_1.jpg', 'face_image_11.jpg',
'face_image_116.jpg', 'face_image_14.jpg', 'face_image_148.jpg',
'face_image_16.jpg', 'face_image_162.jpg', 'face_image_172.jpg',
'face_image_177.jpg', 'face_image_184.jpg', 'face_image_196.jpg',
'face_image_2.jpg', 'face_image_23.jpg', 'face_image_36.jpg', 'face_image_46.jpg',
'face_image_48.jpg', 'face_image_5.jpg', 'face_image_55.jpg', 'face_image_56.jpg',
'face_image_61.jpg', 'face_image_66.jpg', 'face_image_79.jpg', 'face_image_88.jpg'],
1], 'face_image_119.jpg': [['face_image_10.jpg', 'face_image_102.jpg',
'face_image_103.jpg', 'face_image_107.jpg', 'face_image_111.jpg',
'face_image_113.jpg', 'face_image_12.jpg', 'face_image_13.jpg',
'face_image_132.jpg', 'face_image_136.jpg', 'face_image_142.jpg',
'face_image_170.jpg', 'face_image_178.jpg', 'face_image_183.jpg',
'face_image_191.jpg', 'face_image_26.jpg', 'face_image_39.jpg', 'face_image_42.jpg',
'face_image_43.jpg', 'face_image_53.jpg', 'face_image_59.jpg', 'face_image_63.jpg',
'face_image_64.jpg', 'face_image_65.jpg', 'face_image_68.jpg', 'face_image_72.jpg',
'face_image_73.jpg', 'face_image_75.jpg', 'face_image_85.jpg', 'face_image_94.jpg',
'face_image_97.jpg'], 5], 'face_image_126.jpg': [['face_image_104.jpg',
'face_image_109.jpg', 'face_image_114.jpg', 'face_image_120.jpg',
'face_image_123.jpg', 'face_image_124.jpg', 'face_image_125.jpg',
'face_image_131.jpg', 'face_image_139.jpg', 'face_image_140.jpg',
'face_image_146.jpg', 'face_image_150.jpg', 'face_image_152.jpg',
'face_image_164.jpg', 'face_image_166.jpg', 'face_image_168.jpg',
'face_image_169.jpg', 'face_image_171.jpg', 'face_image_174.jpg',
'face_image_176.jpg', 'face_image_179.jpg', 'face_image_182.jpg',
'face_image_189.jpg', 'face_image_193.jpg', 'face_image_21.jpg',
'face_image_22.jpg', 'face_image_24.jpg', 'face_image_25.jpg', 'face_image_29.jpg',
'face_image_3.jpg', 'face_image_35.jpg', 'face_image_41.jpg', 'face_image_51.jpg',
'face_image_71.jpg', 'face_image_78.jpg'], 4], 'face_image_600.jpg':
[array([0.63621665]), 'None'], 'face_image_97.jpg': [['face_image_10.jpg',
'face_image_102.jpg', 'face_image_103.jpg', 'face_image_107.jpg',
'face_image_111.jpg', 'face_image_113.jpg', 'face_image_12.jpg',
'face_image_13.jpg', 'face_image_132.jpg', 'face_image_136.jpg',
'face_image_142.jpg', 'face_image_170.jpg', 'face_image_178.jpg',
'face_image_183.jpg', 'face_image_191.jpg', 'face_image_26.jpg',
'face_image_39.jpg', 'face_image_42.jpg', 'face_image_43.jpg', 'face_image_53.jpg',
'face_image_59.jpg', 'face_image_63.jpg', 'face_image_64.jpg', 'face_image_65.jpg',
'face_image_68.jpg', 'face_image_72.jpg', 'face_image_73.jpg', 'face_image_75.jpg',
'face_image_85.jpg', 'face_image_94.jpg', 'face_image_97.jpg'], 5]}
```

(i)

Based on the idea of face embeddings, I will use bounding box(sliding window with normalized correlation detection) to find all the faces in these pictures and make them an image retrival system (as in (c) –> (f), compute KMeans and build inverted_index and so on..). Then take a picture for every employee, and use these pictures and the retrival system to do matching-process, like in (g) and (h). The challenges might be:

- There are many employees so the clustering process will be slow and not so accurate.
- Also, the increasing number of employees will also slow down the matching process, so the efficient is the main issue here.
- For many images, it is also hard to define a good threshold to filter the similarity scores.
- Different poses of different people might be difficult to detect and cluster.

Question2

(a)

Vanishing gradient is a problem in training the gradient-based neural nets with backpropagation. The problem is when in training process, some gradient will be too small, and this will slow or even stop the updating of corresponding weights, or even stop training of the whole neural nets.

To mitigate vanishing gradient, we can:

- Use a residual network
- Choose activation functions such as ReLU.

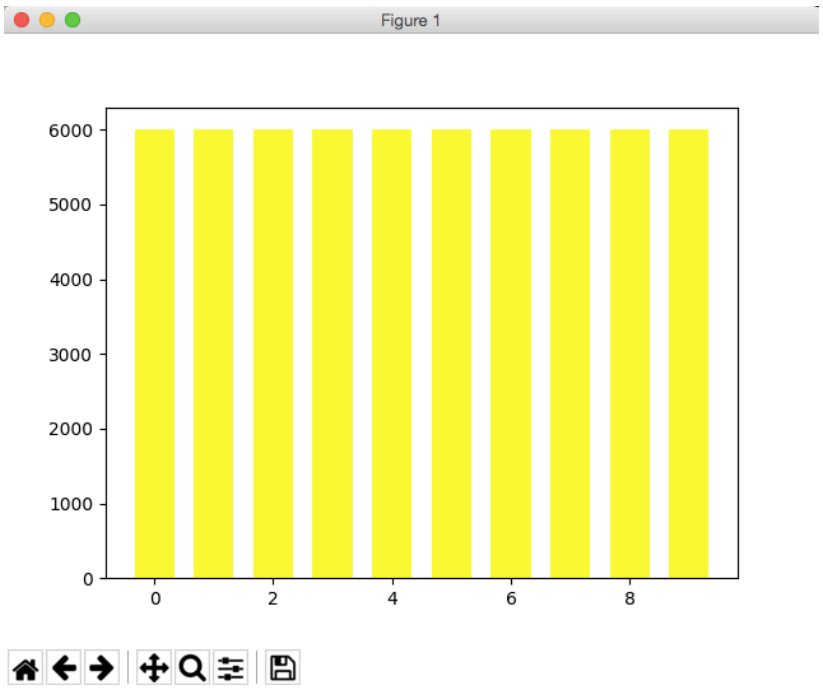
(b)

Max-pooling layer is to down-sample the input, such as reduce its dimensions, so that reduce the computational cost, and can also extract some features from a smaller sub-region of the input.

(c)

(split part in the model(validation_split = 0.3)..)

bar graph:



number of instance = 6000

x-axis: 0-9 represents different kinds of instances

(d)

(in Q2.py)

(e)

categorical_crossentropy

$L_i = -\sum t_{i,j} \log p_{i,j}$

Where **t** is the target(ground truth label) and **p** is the prediction, **i** is the index of data point and **j** is the index of the class.

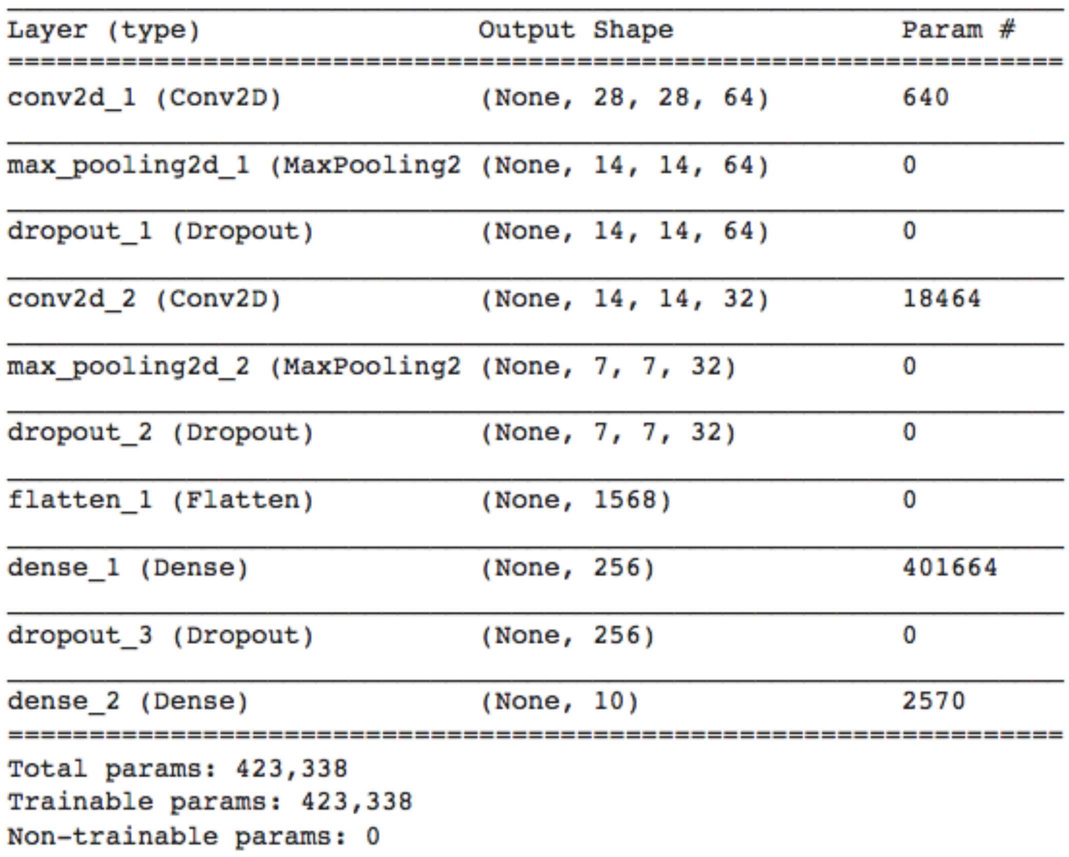
(f)

batch_size = 32

epochs = 20

Should stop training when validation loss significantly increases, because this means our model is going to be overfitting. This is called “early stopping”. In the code we can actually add a “callback” with “patience” to wait for a few more updates to see if the validation loss is just slightly turbulence or significantly getting higher.

Final architecture:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 64)        640       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 64)        0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 32)        18464     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 32)          0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 7, 7, 32)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 1568)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 256)               401664    
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570      
=================================================================
Total params: 423,338
Trainable params: 423,338
Non-trainable params: 0
```

batch_size = 32

epochs_trained = 20

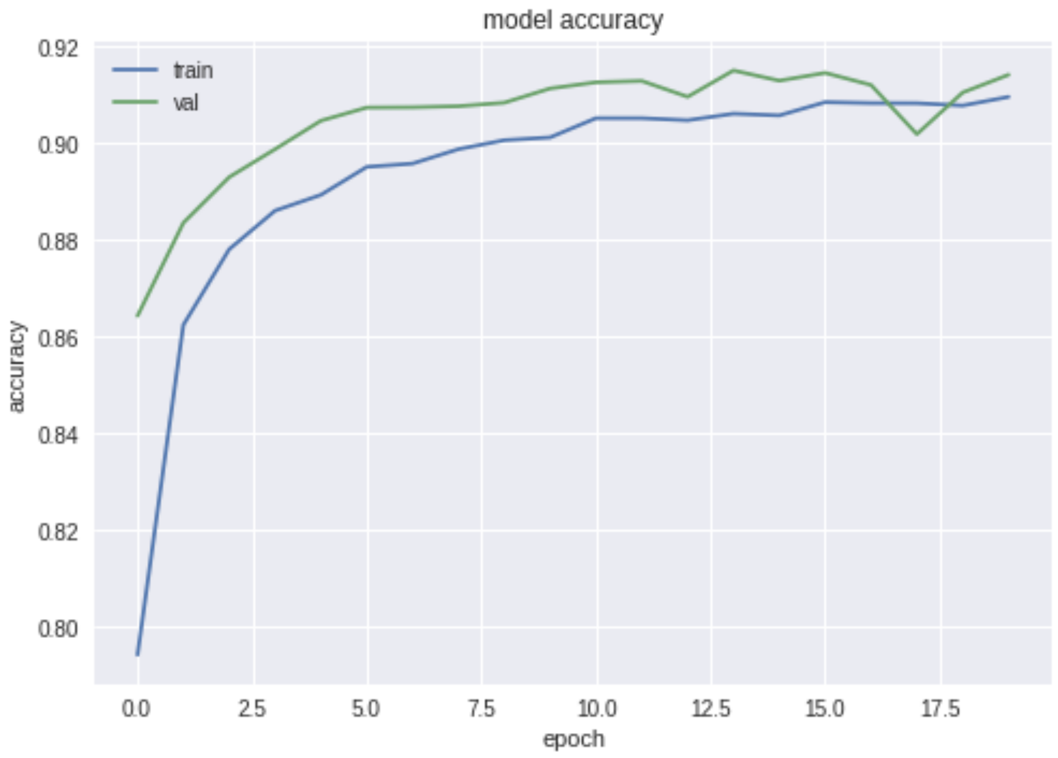
learning rate = 0.0001 (the default setting for Adam optimizer)

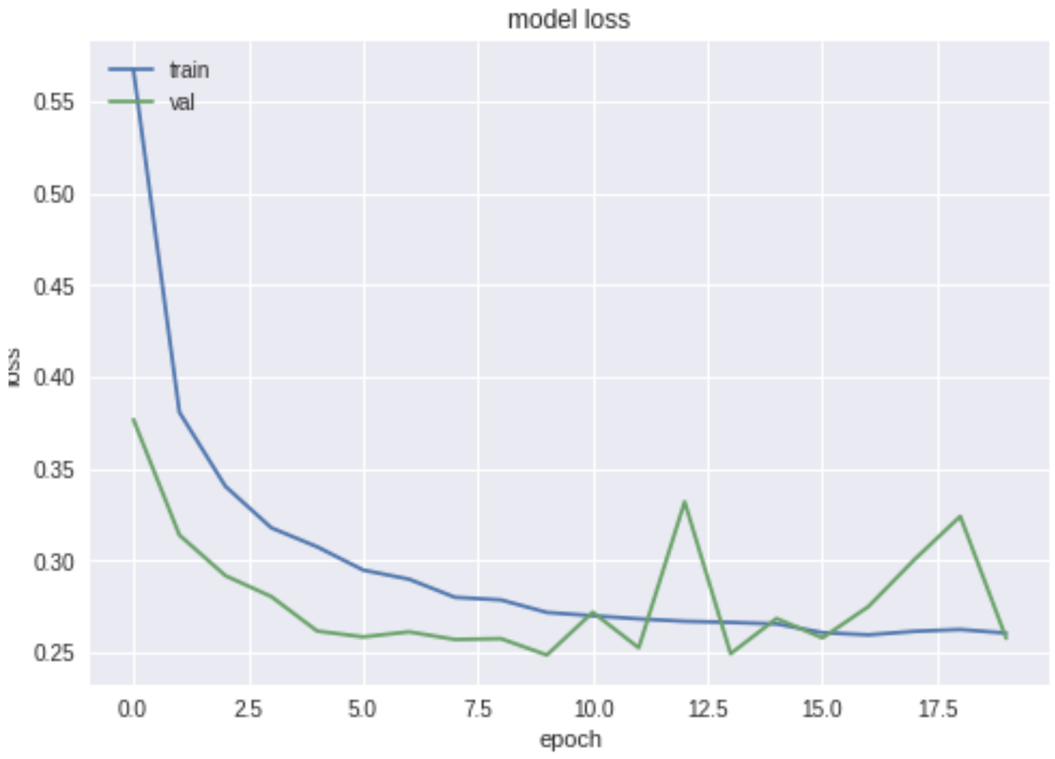
training_loss = 0.2606

validation_loss = 0.2579

training_acc = 0.9096

validation_acc = 0.9142

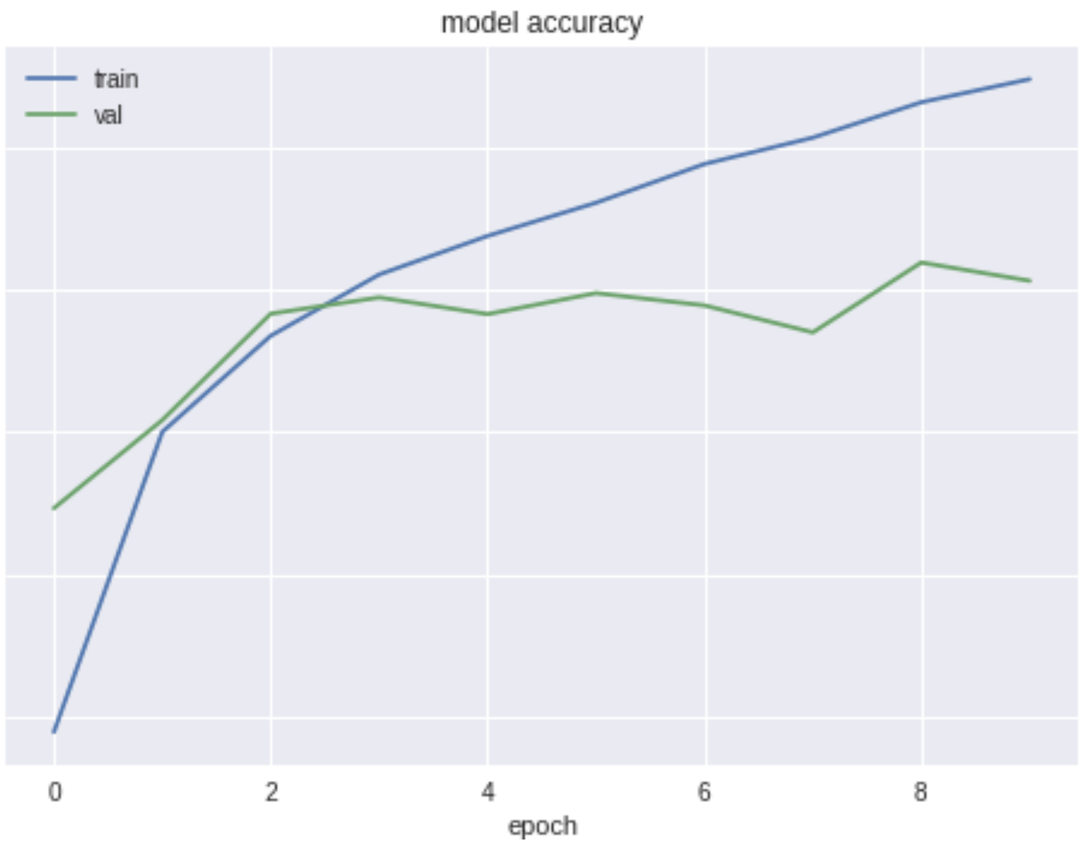


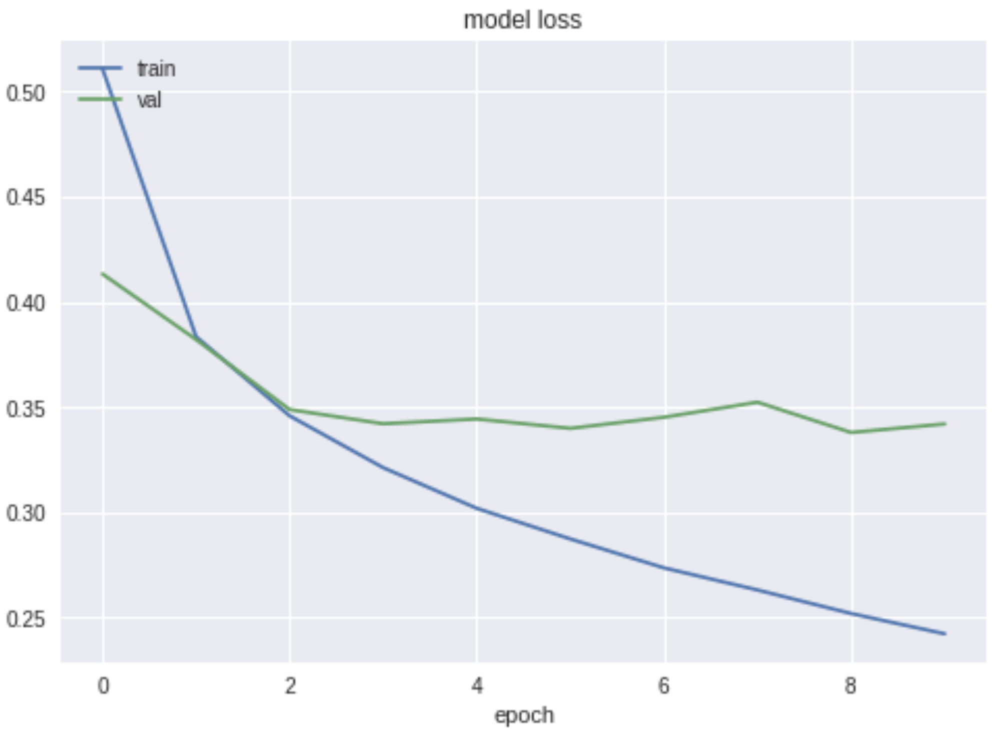


Final Test acc = 0.9054

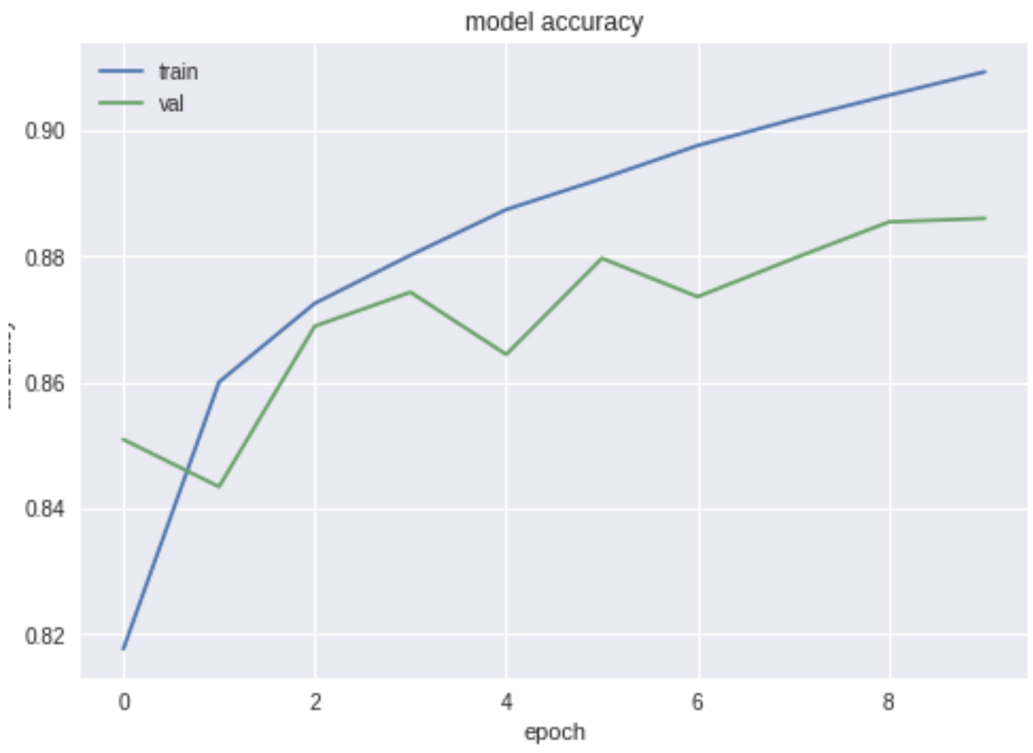
(g)

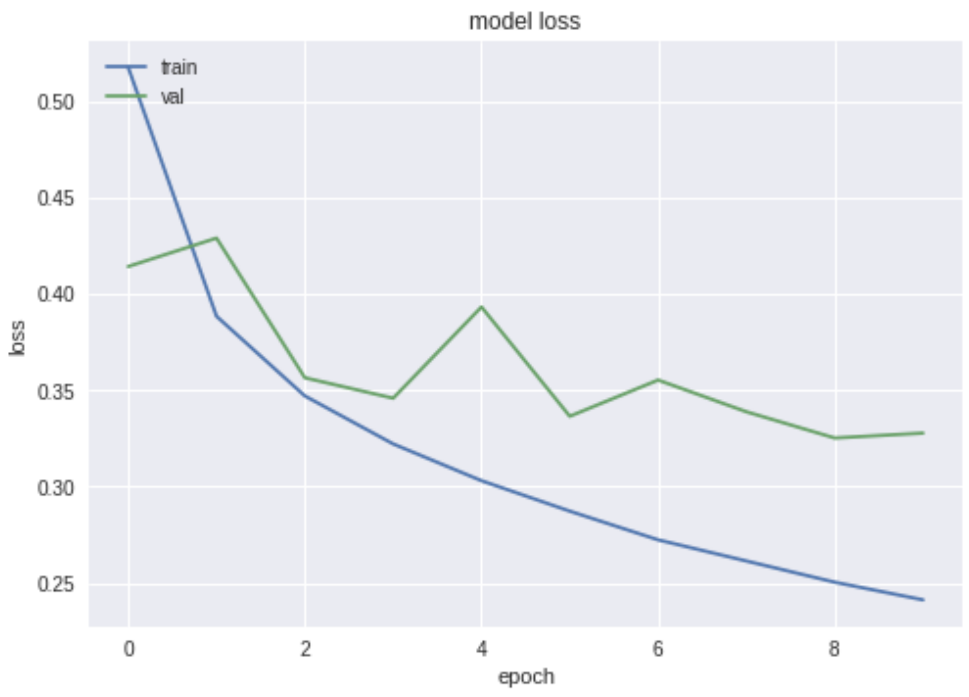
batch size = 8:



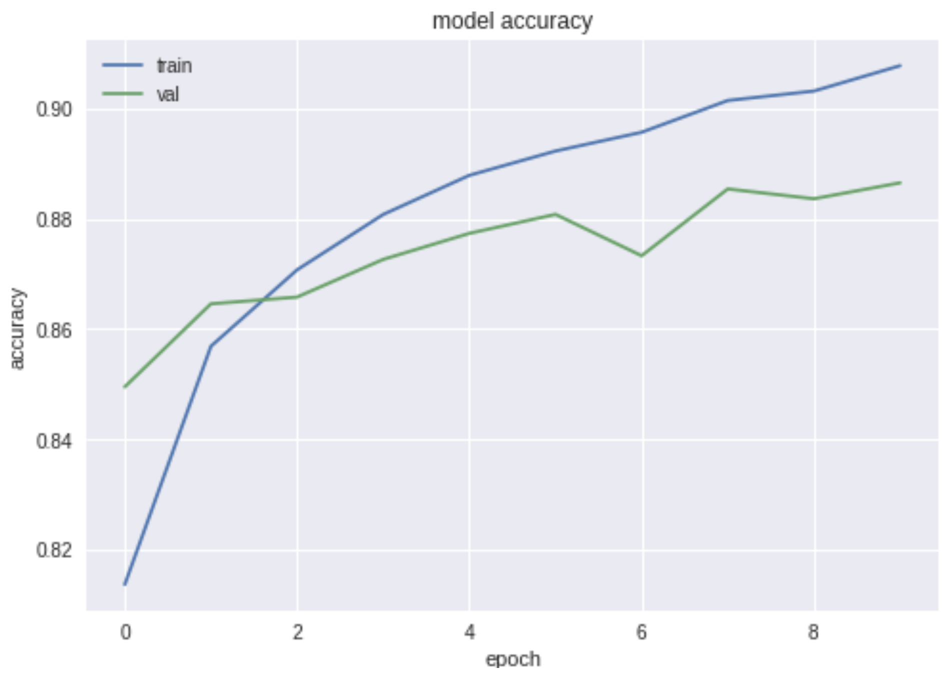


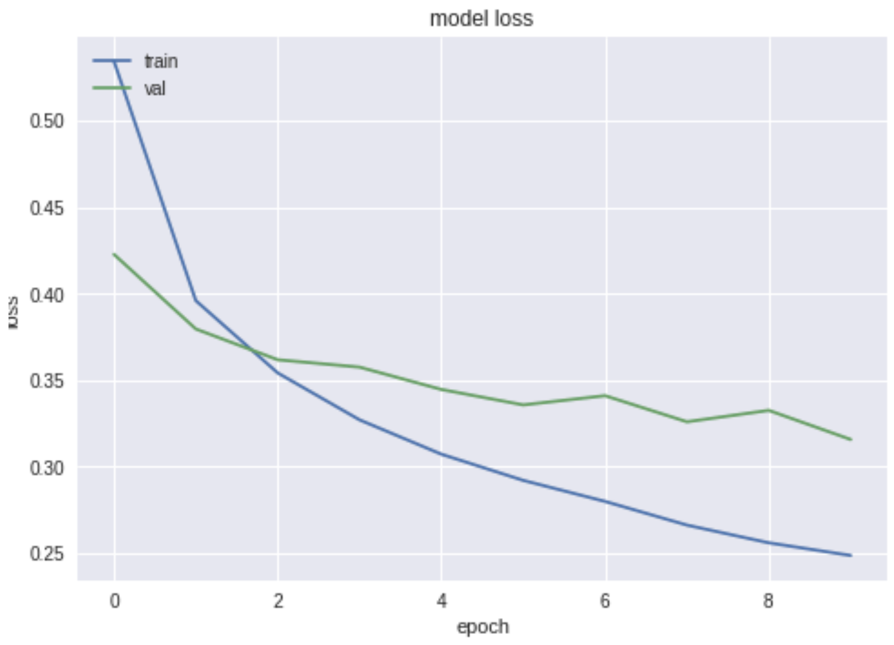
batch_size = 16:



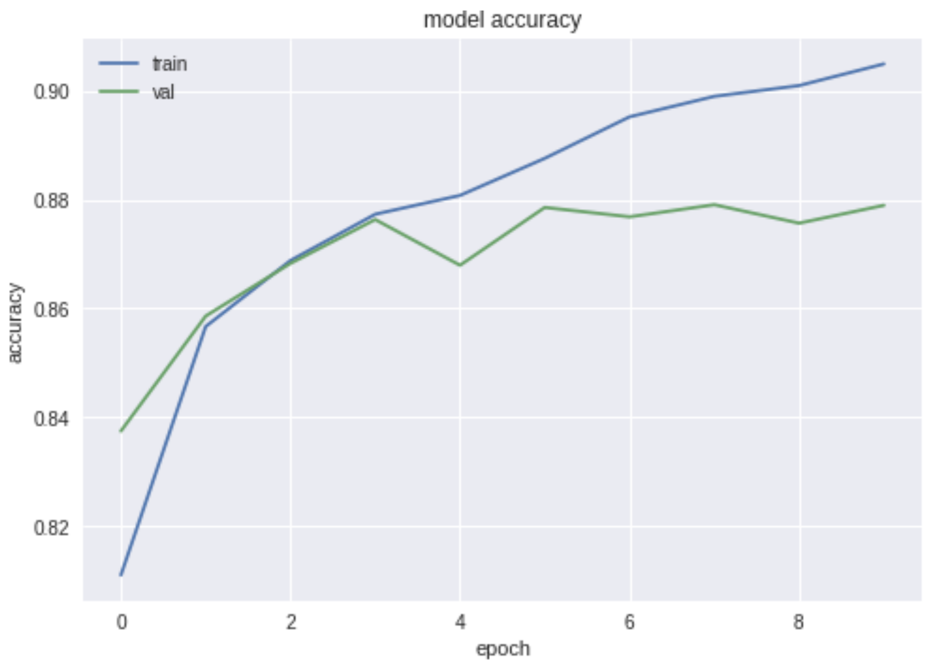


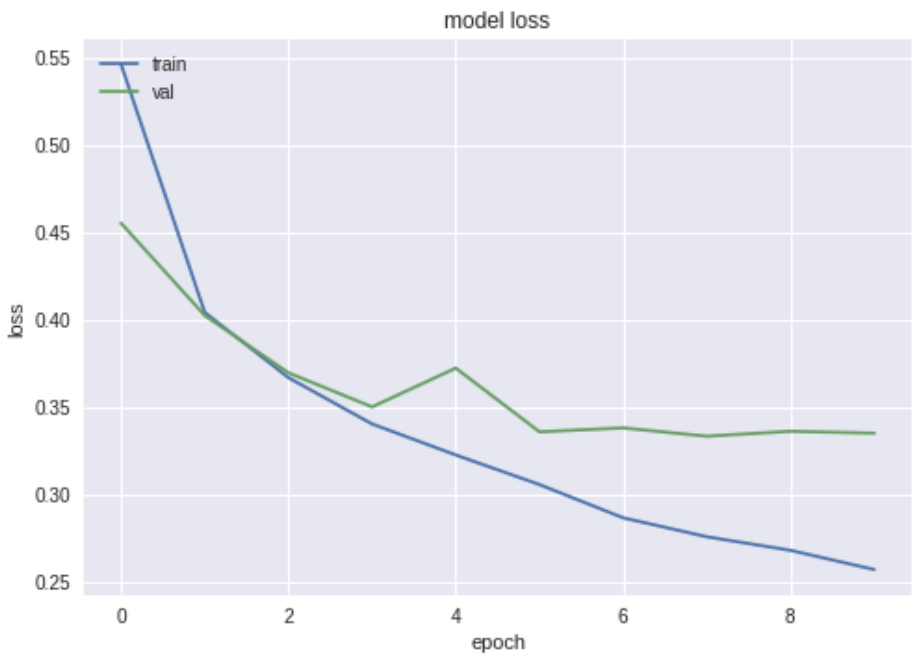
batch_size = 32:





batch_size = 64:





Large batch size means we averaged the variance more, so the validation loss curve will be more stable when the batch size increases. And also bigger batch size will use fewer updates to achieve the same accuracy, i.e., faster to converge.