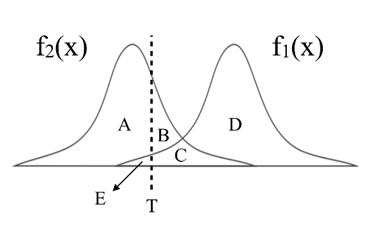
Computer Vision: from Recognition to Geometry

HW2

Name: 黃沛沅 Department: 機械系 Student ID: B04605067

Problem 1

1. Assume 𝑋 is a continuous random variable that denotes the estimated probability of a binary classifier. The instance is classified as positive if 𝑋 > 𝑇 and negative otherwise. When the instance is positive, 𝑋 follows a PDF 𝑓1(𝑥). When the instance is negative, 𝑋 follows a PDF 𝑓2(𝑥). Please specify which regions (A ~ E) represent the cases of *False Positive* and *False Negative*, respectively. Clearly explain why. (6%)



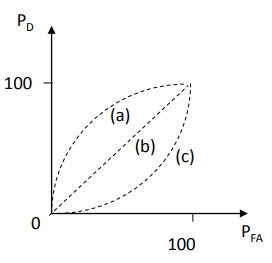
False positive means that the testing result present positive (X>T), while it is negative in reality;

False negative means that the testing result present negative(X<T), while it is positive in reality.

B, C represent the case of false positive.

E represents the case of false negative.

1. There are three ROC curves in the plot below. Please specify which ROC curves are considered to have reasonable discriminating ability, and which are not. Also, please answer that under what circumstances will the ROC curve fall on curve (b)? (6%)



(a) curve is the reasonable one, and (c) is not. As the professor mentioned in class while we move the threshold T from negative infinity toward positive direction, the value of PFA drops faster than the value of PD. So, (a) is reasonable, and (c) is impossible.

The ROC will fall on curve (b) if the f1 and f2 have the same distribution. Therefore, the PD and PFA are always the same no matter where we put the threshold.

Problem 2

1. Given a convolutional layer which contains:   
    𝑛 kernels with size   
    padding size 𝑝,   
    stride size (𝑠,𝑠).   
   With input feature size , calculate the size of output feature .
   1. 𝑊\_𝑜𝑢𝑡=? (2%)
   2. 𝑛\_𝑜𝑢𝑡 =? (2%)

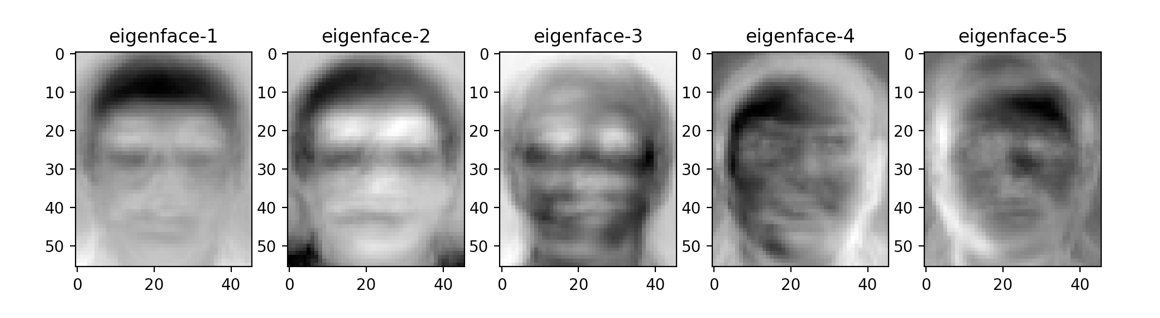
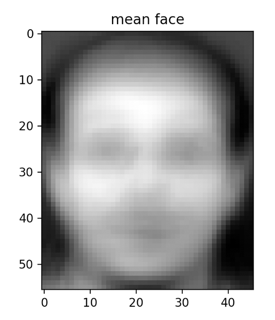
1. 𝑘=5, 𝑠=2, 𝑝=1, 𝑛=256, 𝑊=64, calculate the number of parameters in the convolutional layer (4%)

Problem 3 (report 35% + code 5%)

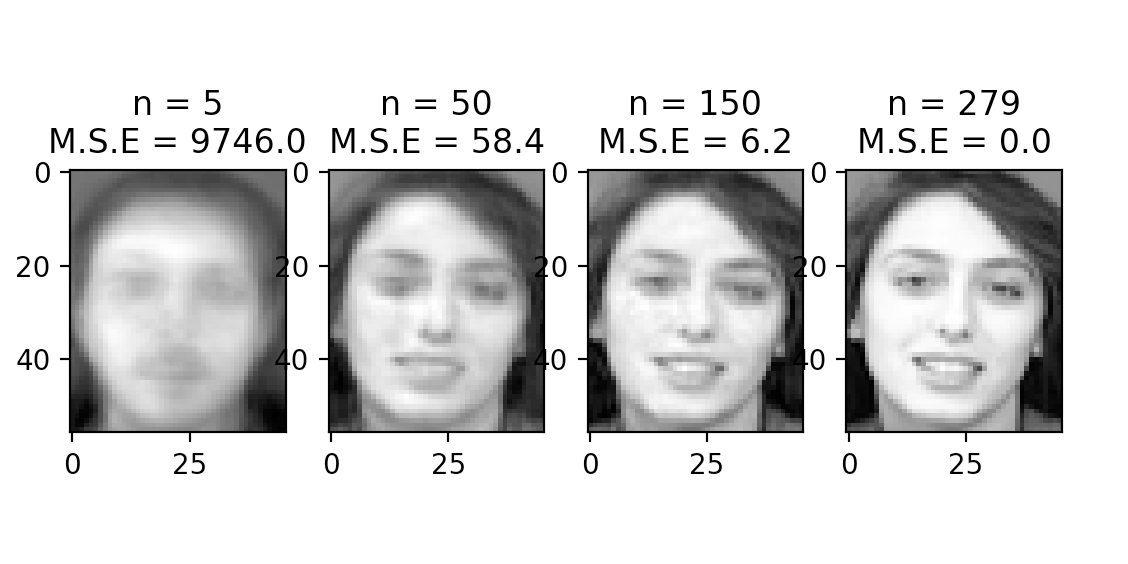
1. PCA (15%)

In this task, you need to implement PCA from scratch, which means you cannot call PCA function directly from existing packages.

1. Perform PCA on the training data. Plot the mean face and the first five eigenfaces and show them in the report. (5%)

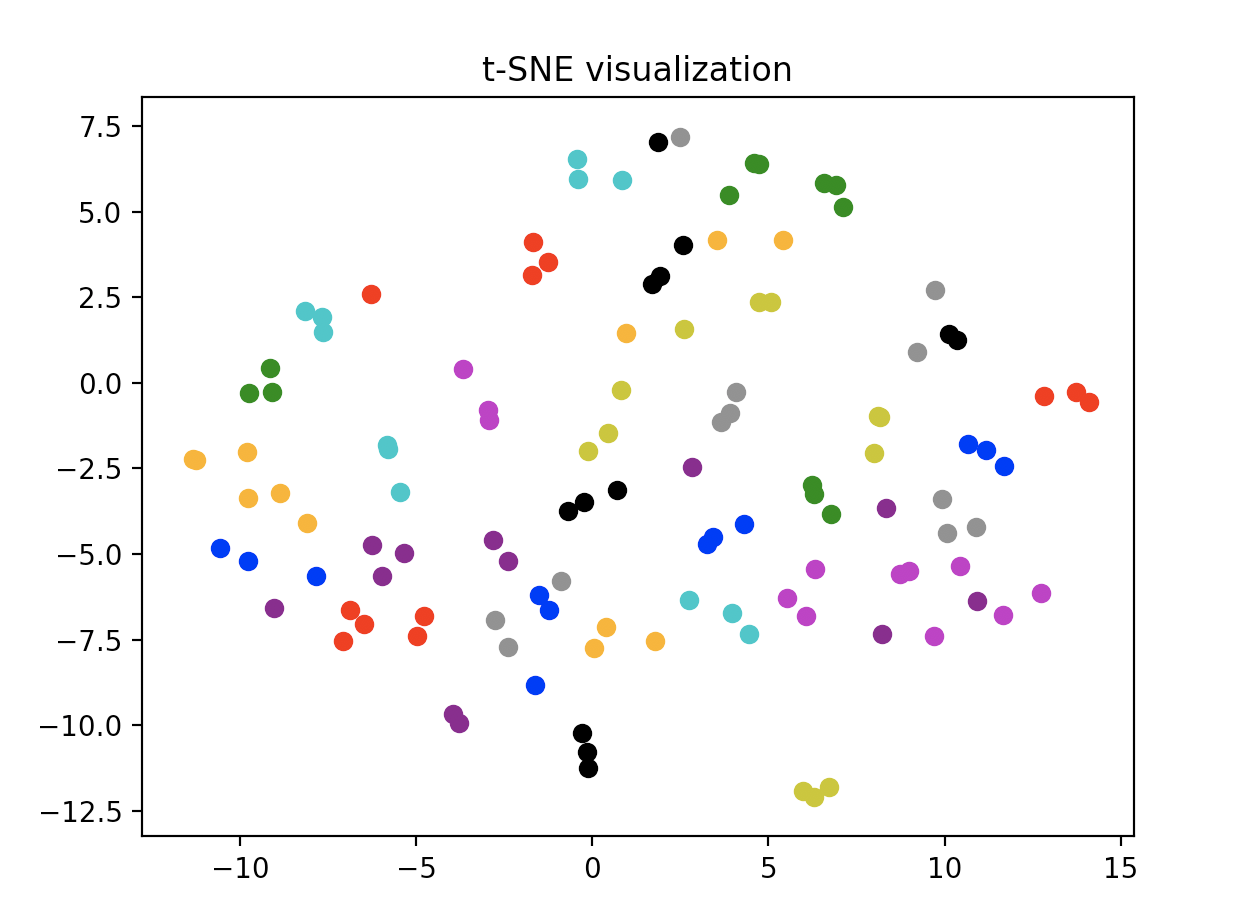


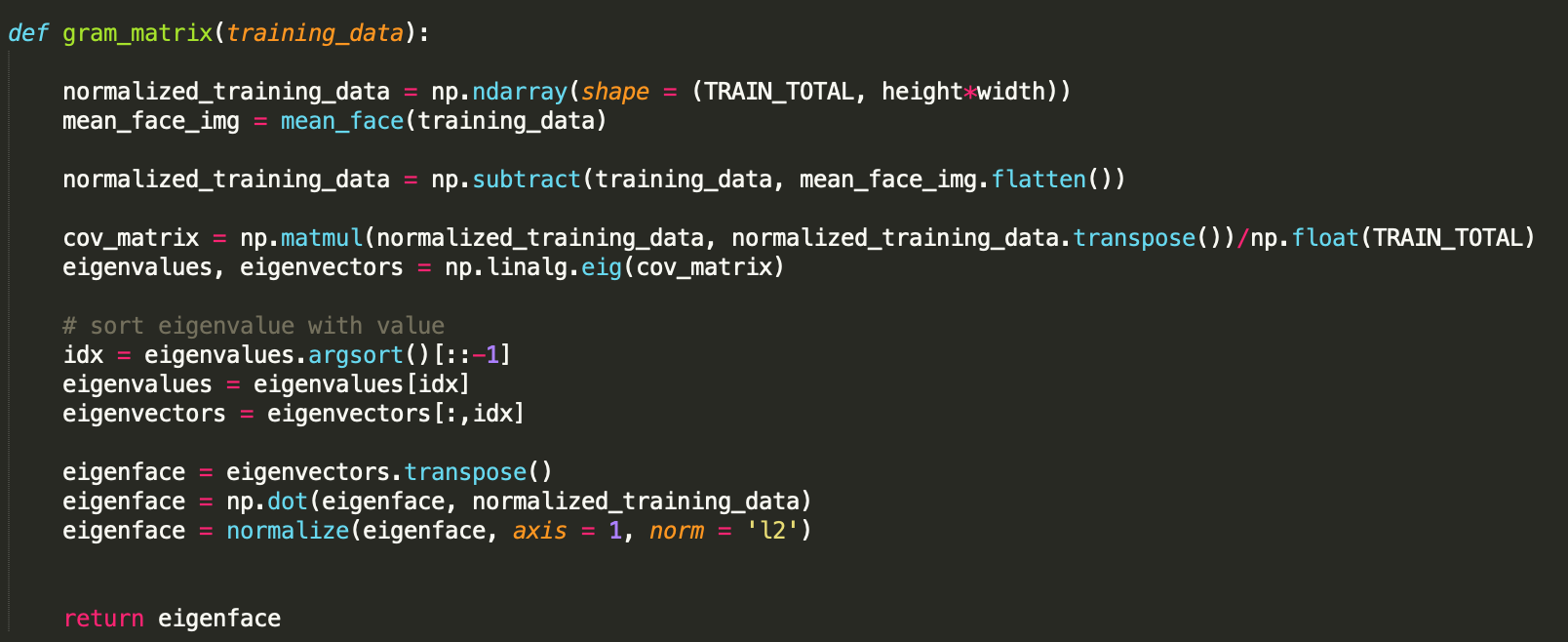
1. Take 𝒑𝒆𝒓𝒔𝒐𝒏8\_ 𝒊𝒎𝒂𝒈𝒆6, and project it onto the above PCA eigenspace. Reconstruct this image using the first n = { 5, 50, 150, all } eigenfaces. For each n, compute the mean square error (MSE) between the reconstructed face image and the original 𝒑𝒆𝒓𝒔𝒐𝒏8\_ 𝒊𝒎𝒂𝒈𝒆6. Plot these reconstructed images with the corresponding MSE values in the report. (5%)



We can see that when we use all eigenfaces, the image M.S.E. is zero.

1. Reduce the dimension of the image in testing set to dim = 100. Use t-SNE to visualize the distribution of test images. (5%)



1. (bonus 5%) Implement the Gram Matrix trick for PCA. Compare the two reconstruction images from standard PCA/Gram Matrix process. If the results are different, please explain the reason. Paste the main code fragment(screenshot) on the report with discussion.

Gram Matrix process takes less time to calculate, and the result is the same as PCA process. It’s because when counting eigenvectors the gram matrix has dimension of 280\*280 while the PCA has 2576\*2576. The result is shown when running the code.

1. k-NN (10%)

To apply the k-nearest neighbors (k-NN) classifier to recognize the testing set images, please determine the best k and n values by 3-fold cross-validation.

For simplicity, the choices for such hyper-parameters are:

k = {1, 3, 5} and n = {3, 10, 39}.

Please show the cross-validation results and explain your choice for (k, n). Also, show the recognition rate on the testing set using your hyper-parameter choice.

The table below shows k, n and their corresponding accuracy

|  |  |  |  |
| --- | --- | --- | --- |
| k n | 3 | 10 | 39 |
| 1 | 0.6833 | 0.8514 | 0.9236 |
| 3 | 0.5931 | 0.7458 | 0.8486 |
| 5 | 0.5153 | 0.6778 | 0.7764 |

The best k, n determined by cross-validation is 1, 39, which use the most eigenvectors and the least reference neighbor.

The accuracy on the testing with (k, n) = (1, 39) is 0.958 .

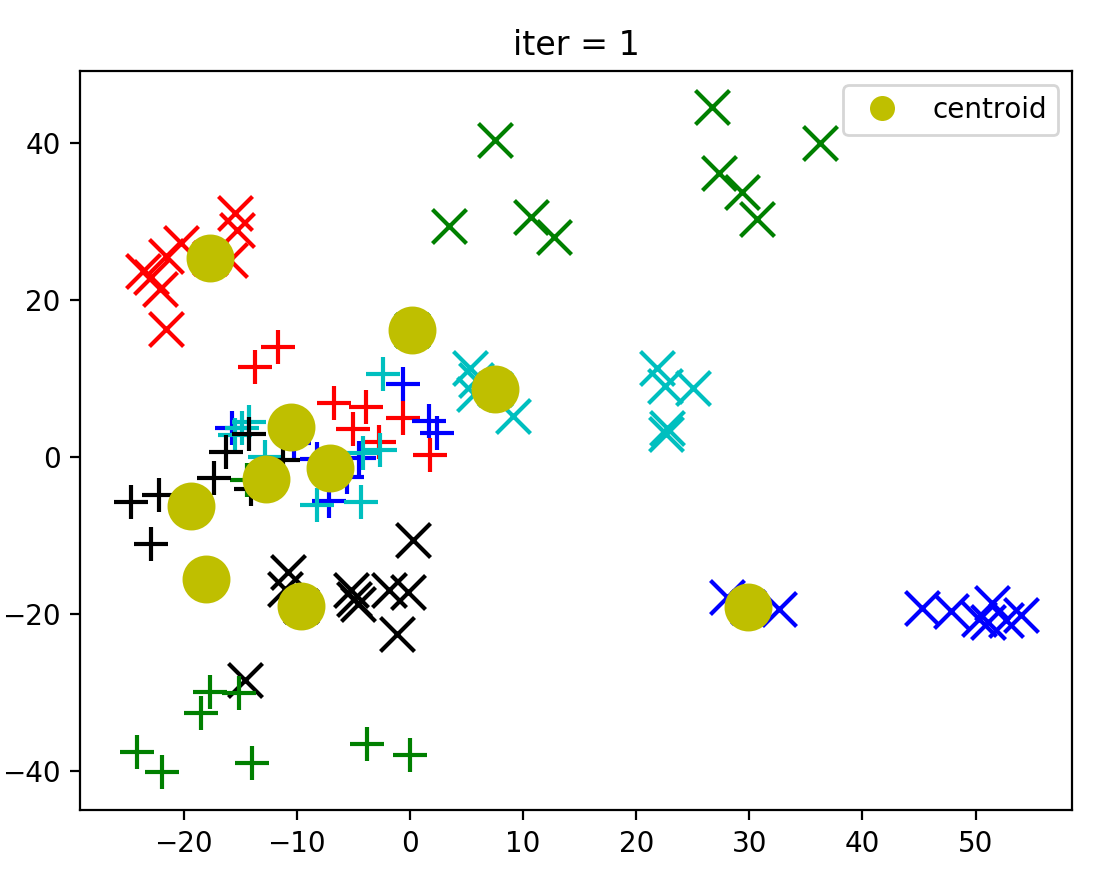
1. K-means (10%)

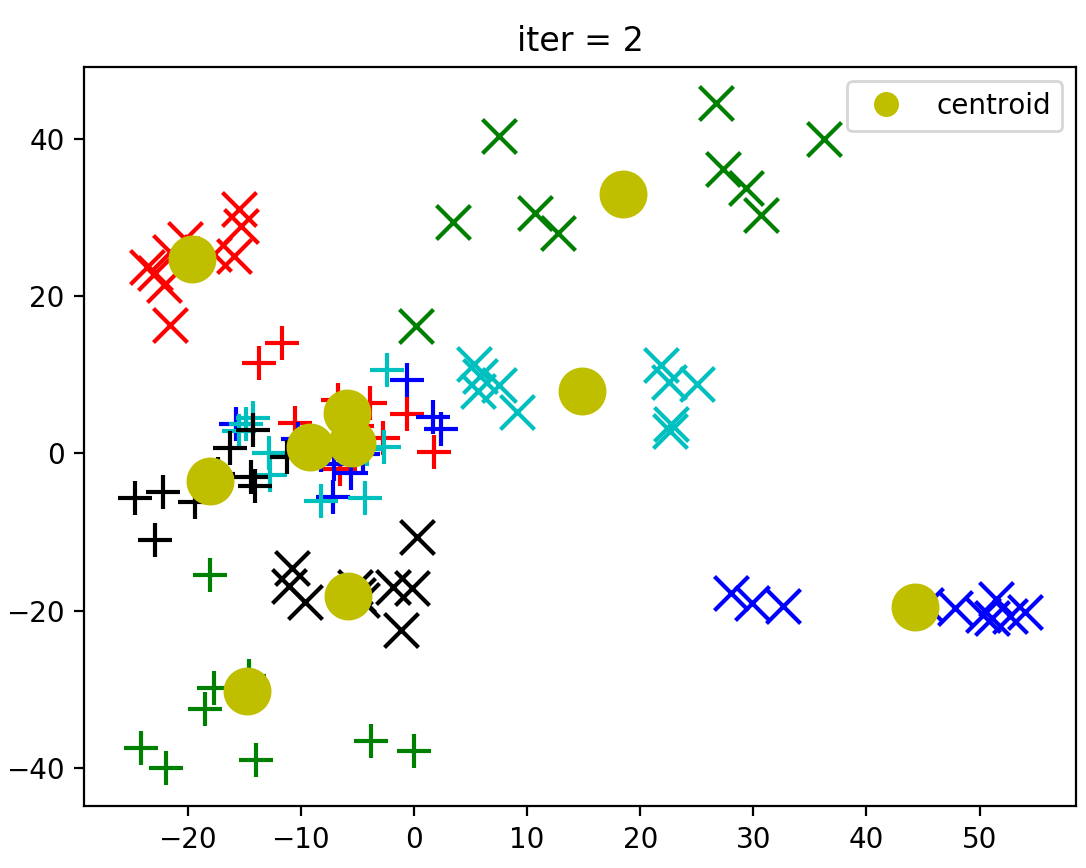
Reduce the dimension of the images in the first 10 class of training set to dim = 10 using PCA. Implement the k-means clustering method to classify the these images.

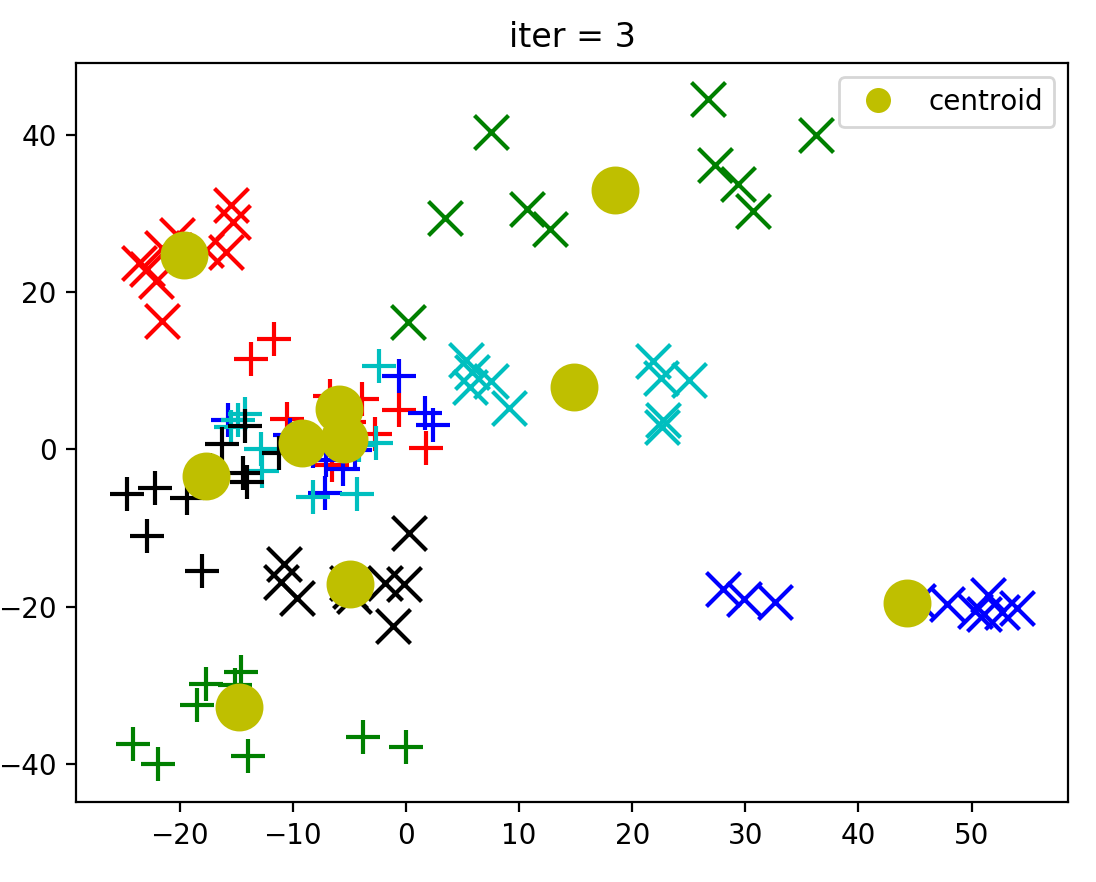
1. Please use weighted Euclidean distance to implement the k-means clustering. The weight of the best 10 eigenfaces is

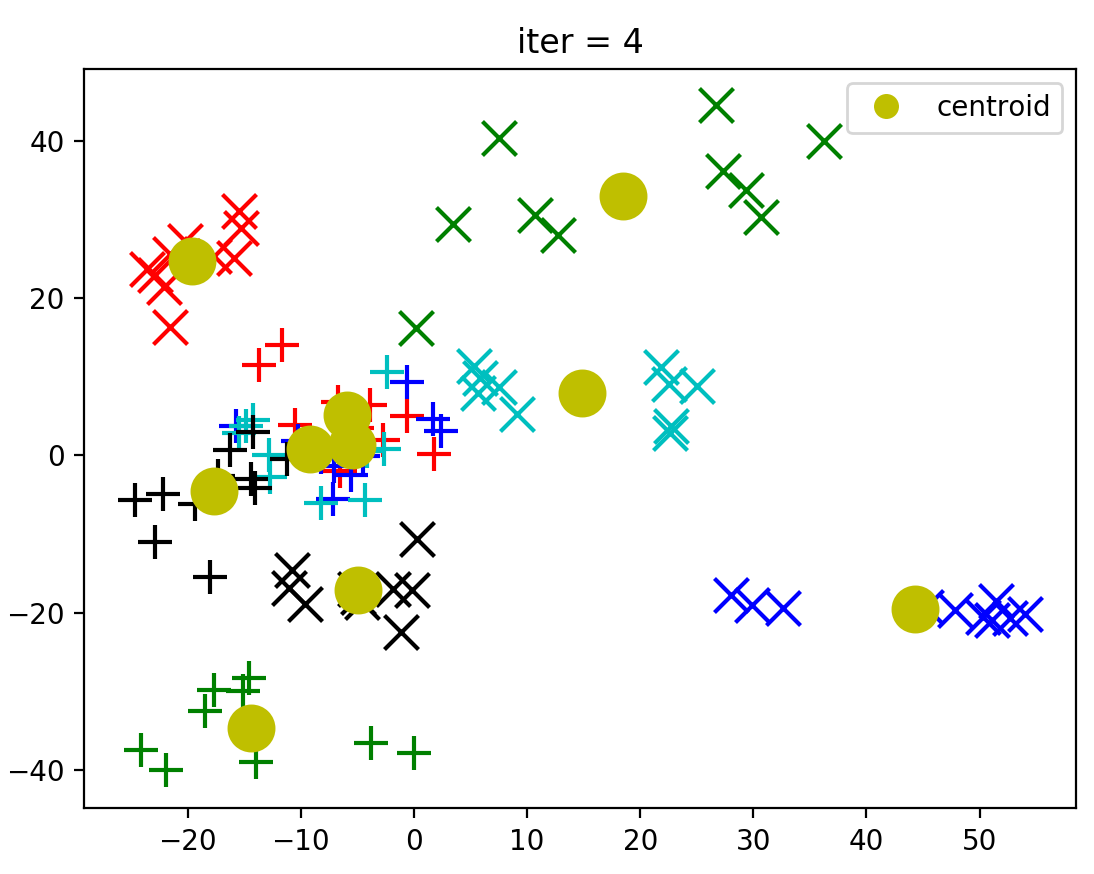
[ 0.6, 0.4, 0.2, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1 ]

1. Please visualize the features and the centroids to 2D space with the first two eigenfaces for each iteration in k-means clustering (up to 5 images). (5%)









1. Compare the results of K-means and ground truth. (5%)

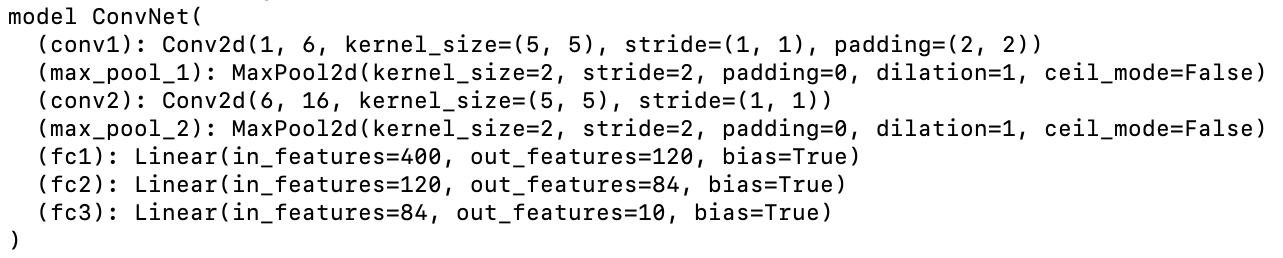
Accuracy = 0.985

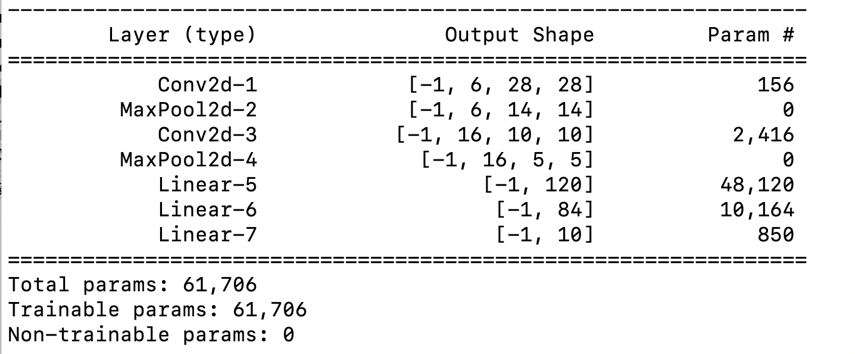
Only the first data is incorrectly predicted as class\_1, and others are truly predicted.

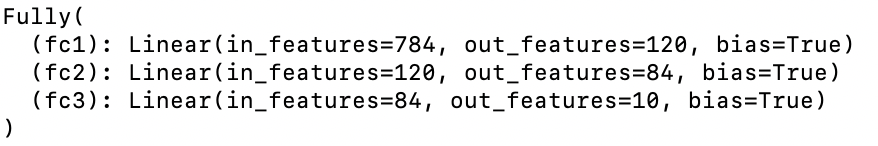
Problem 4 (report 30% + baseline 10%)

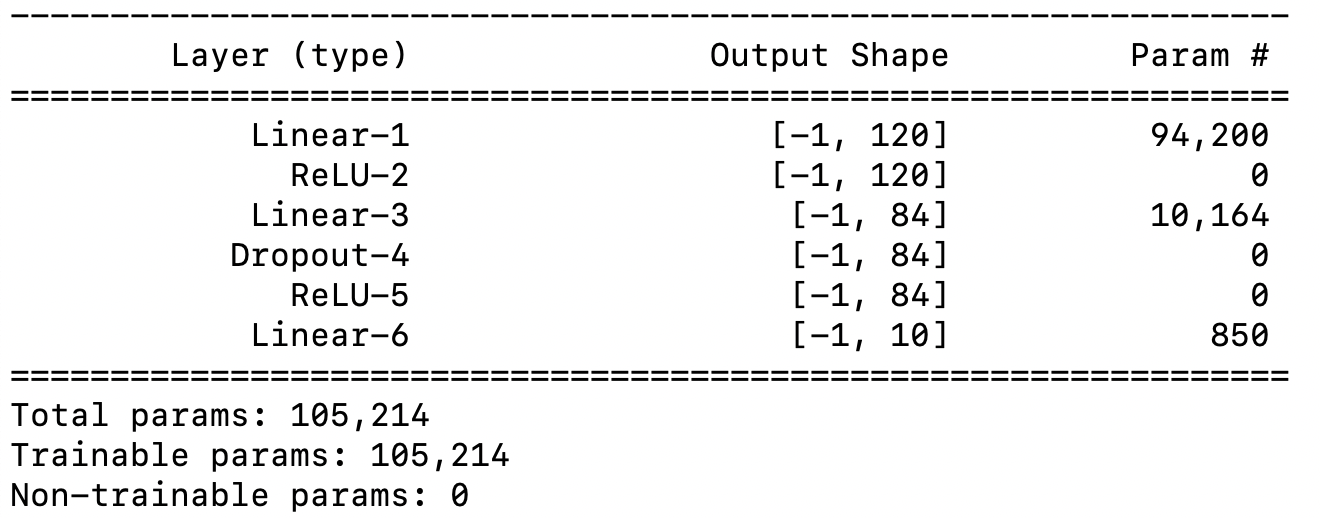
1. MNIST Classification (20%)
2. Build a CNN model and a Fully-Connected Network and train them on the MNIST dataset. Show the architecture of your models and the correspond parameters amount in the report. (5%)

The convolution NN used totally 61706 parameters, while the fully connected network used 105214 parameters. We can see that CNN used much fewer parameters than fully connected network did.





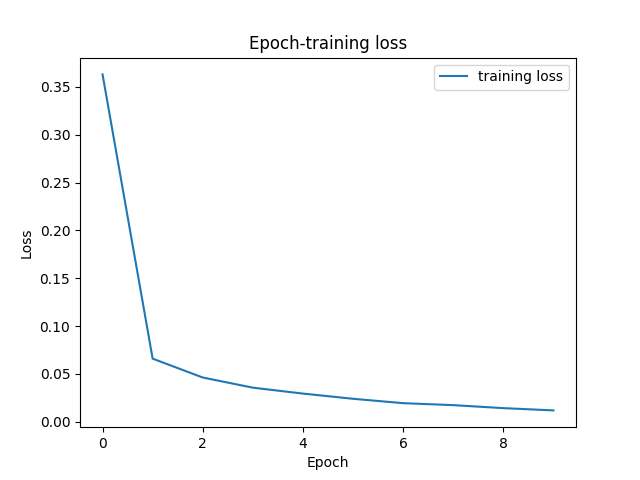
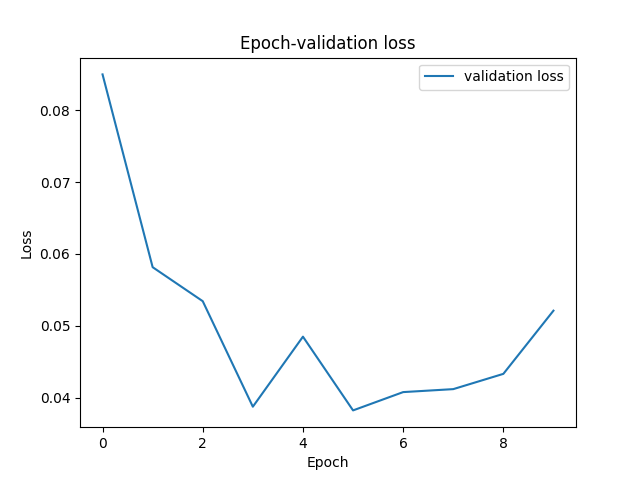


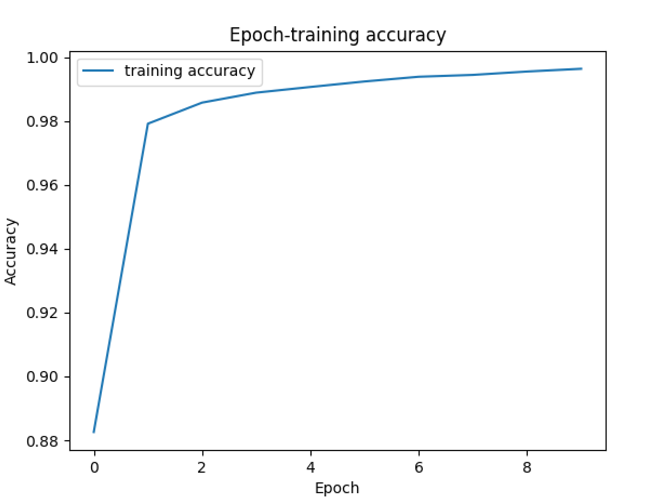
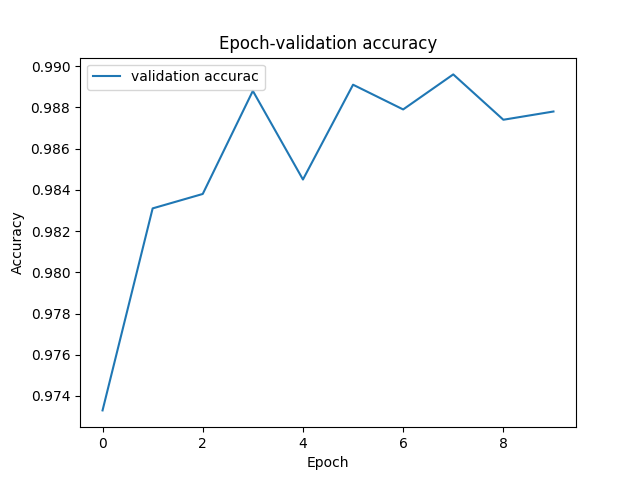


1. Report your training / validation accuracy, and plot the learning curve (loss, accuracy) of the training process. (figure 5%+ baseline 5%)

Lenet-5 model training process

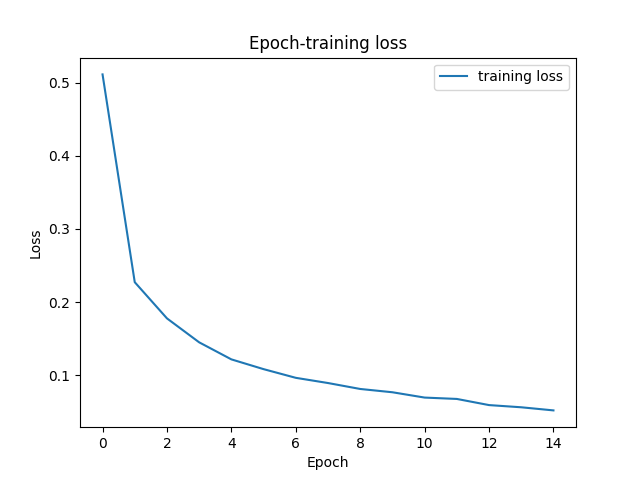
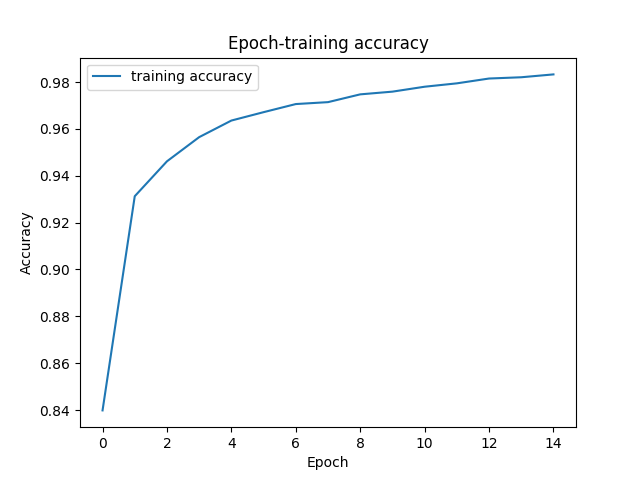
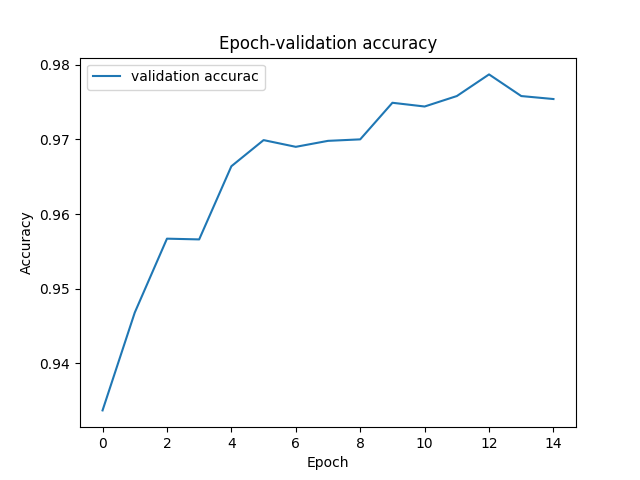
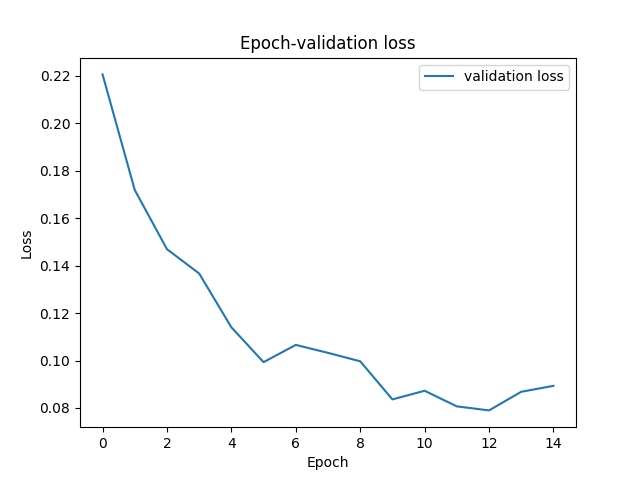
Training accuracy: 0.996

Validation accuracy: 0.99



Fully connected network training process

Training accuracy: 0.988

 Validation accuracy: 0.98

1. Compare the results of both models and explain the difference. (5%)

CNN used takes less computation cost but perform better than FCN. The reason probably is that CNN used more efficient way to learn. In convolutional layer in CNN, the network got image features first and used less dimension vector to present it. Then when it went to fully connected layer, it left fewer parameters but useful vectors as features. Therefore, its time costing is low, but performance is good.

1. Face Recognition (20%)
2. Extract image feature using pytorch pretrained alexnet and train a KNN classifier to perform human face recognition on the given dataset. Report the validation accuracy. (5%)

Accuracy: 0.99

Validation accuracy: 0.61

I take n neighbor to be 1, and don’t do PCA dimension reduction because after doing PCA, I find that the validation accuracy decreases not few.

1. Build your own model and train it on the given dataset to surpass the accuracy of previous stage. Show and explain the architecture of your model and report the validation accuracy. Paste the main code fragment(screenshot). (5%)

Validation accuracy: 0.915

I use resnet18 using the code similar to some website resources. Resnet18 includes convolution layer, batch normalization, maxpool, basic blocks, average pool and fully connected layers.

The image initially goes into convolution layer, and then be normalized by batch normalization layer. The batch normalization normalizes the features without changing its dimension and it improve the efficiency. Before going Resnet layer, it is maxpooled.

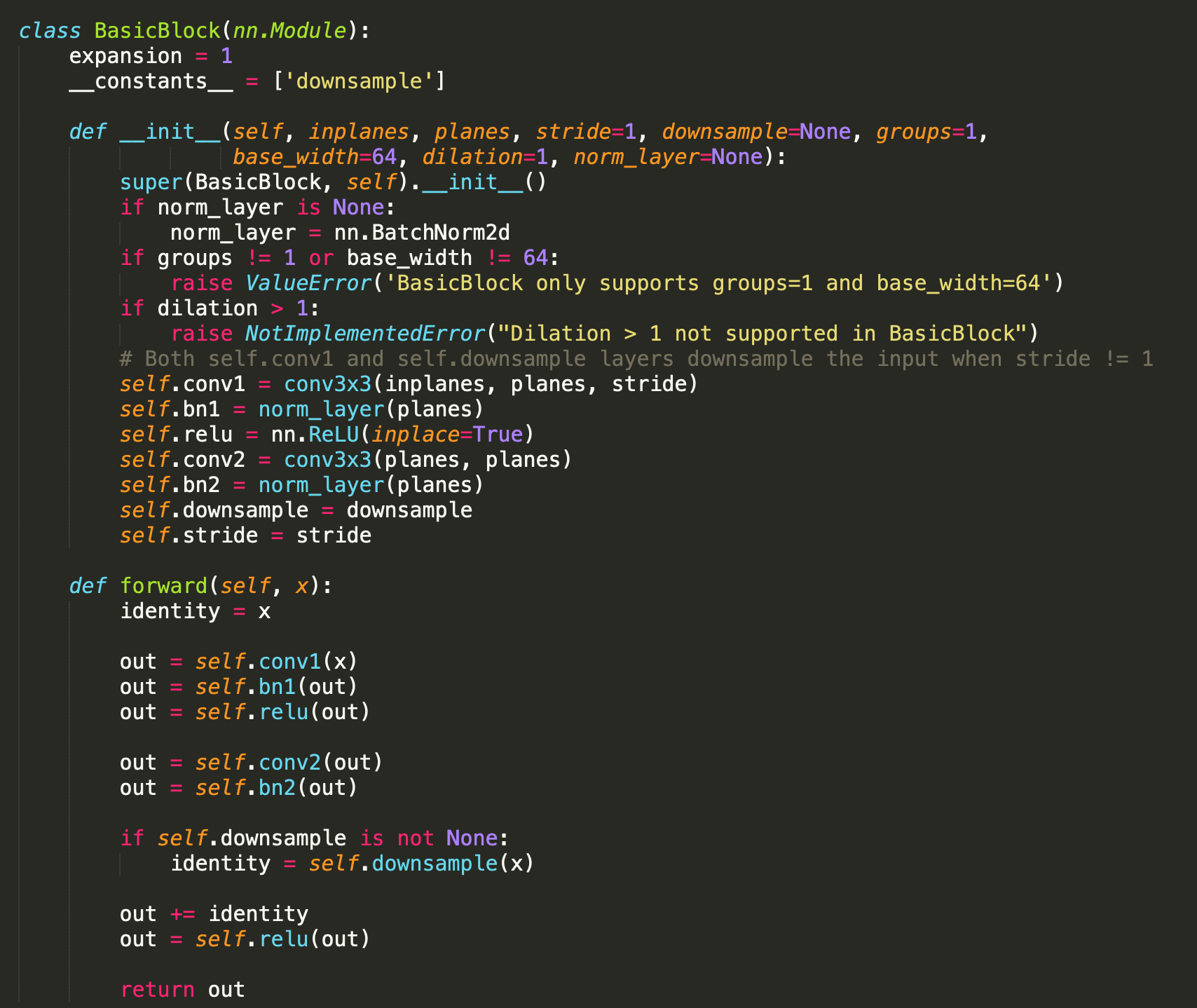
Every layer of Resnet is composed of two blocks.

In Resnet18, each layer has two basic blocks, containing conv, bn1, relu, conv, bn2, downsampling.

The down sampling of the volume through the network is achieved by increasing the stride, making it fit the size we want.

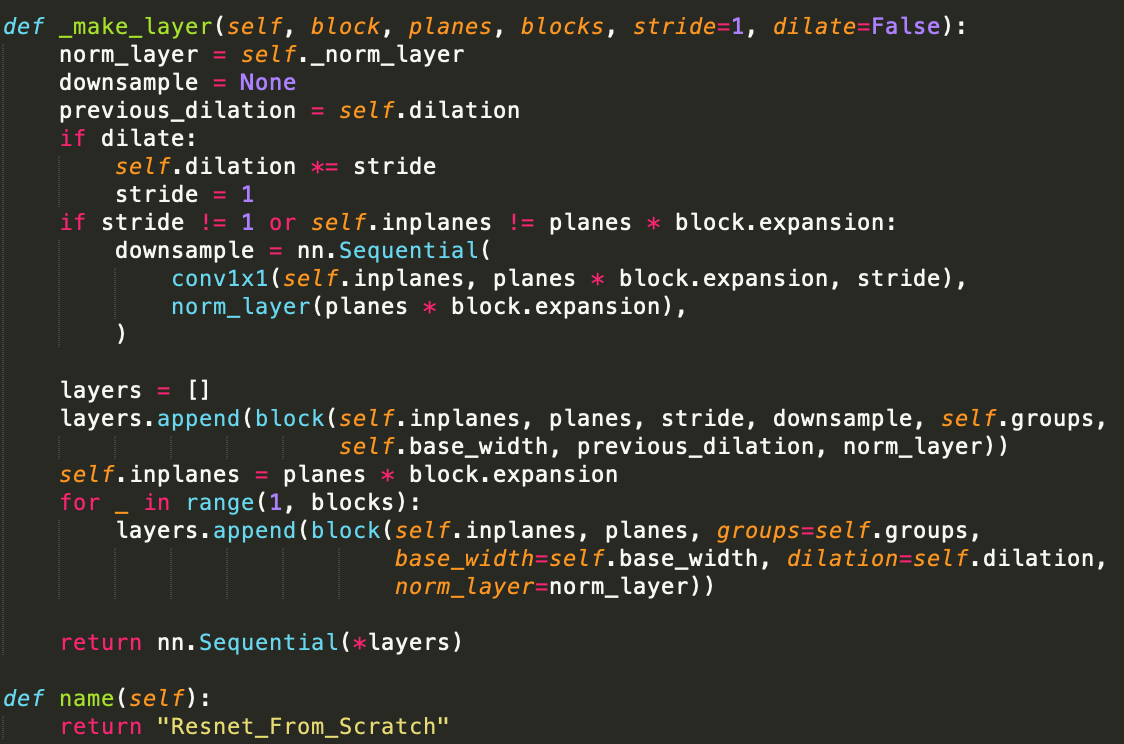
Finally, it turns out to be the prediction result by passing through average pooling and fully connected network.

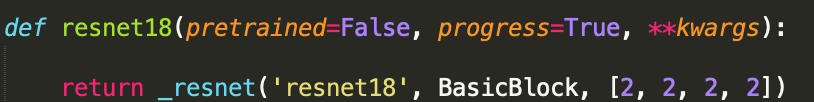
Basic Block



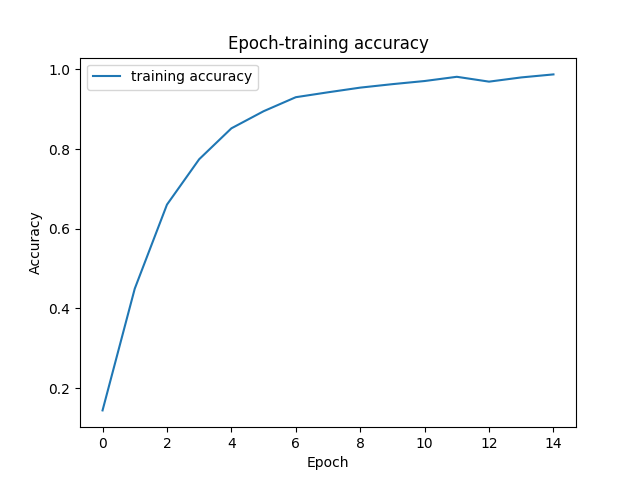
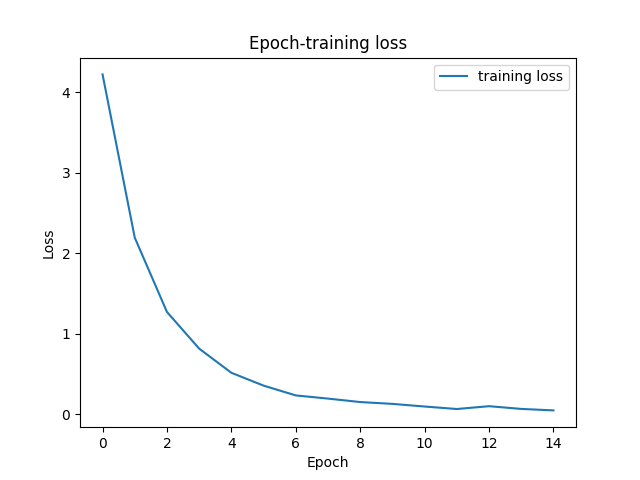
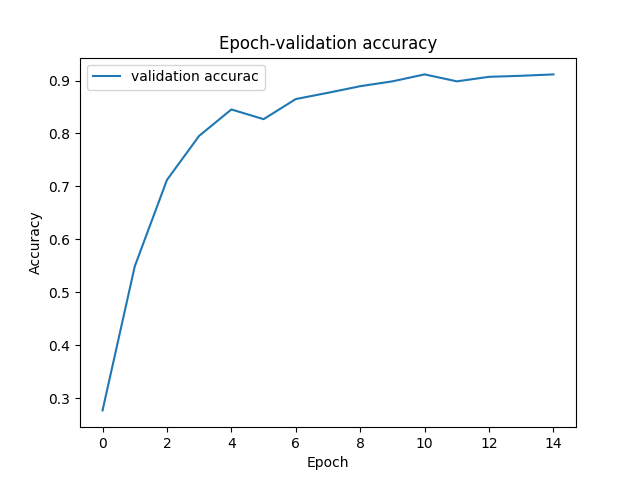
Resnet structure

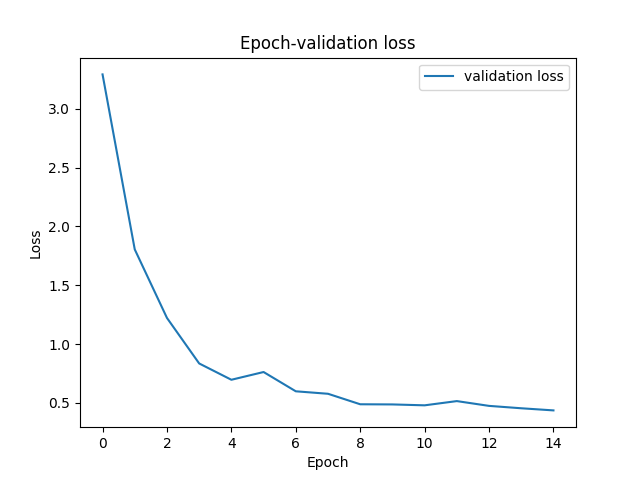






1. Plot the learning curve of your training process (training/validation loss/accuracy) in stage 2, explain the result you observed. (5%)

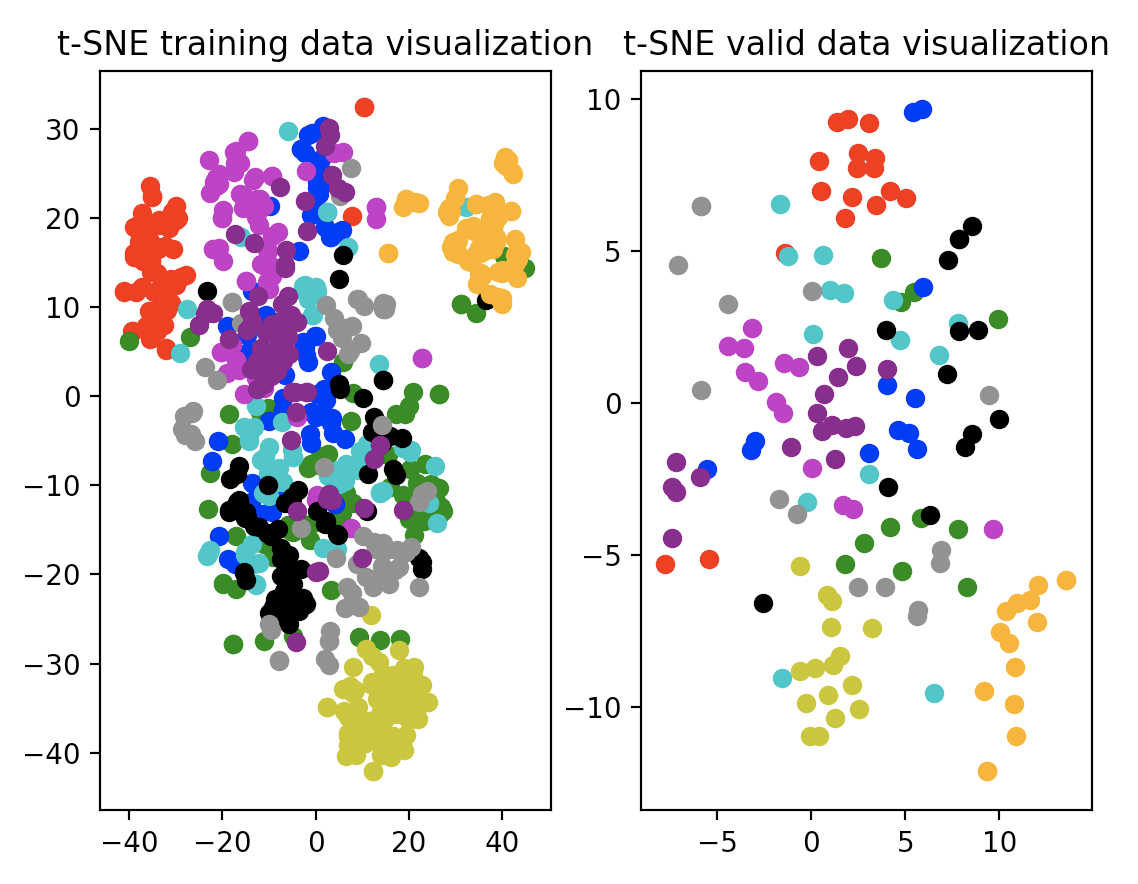




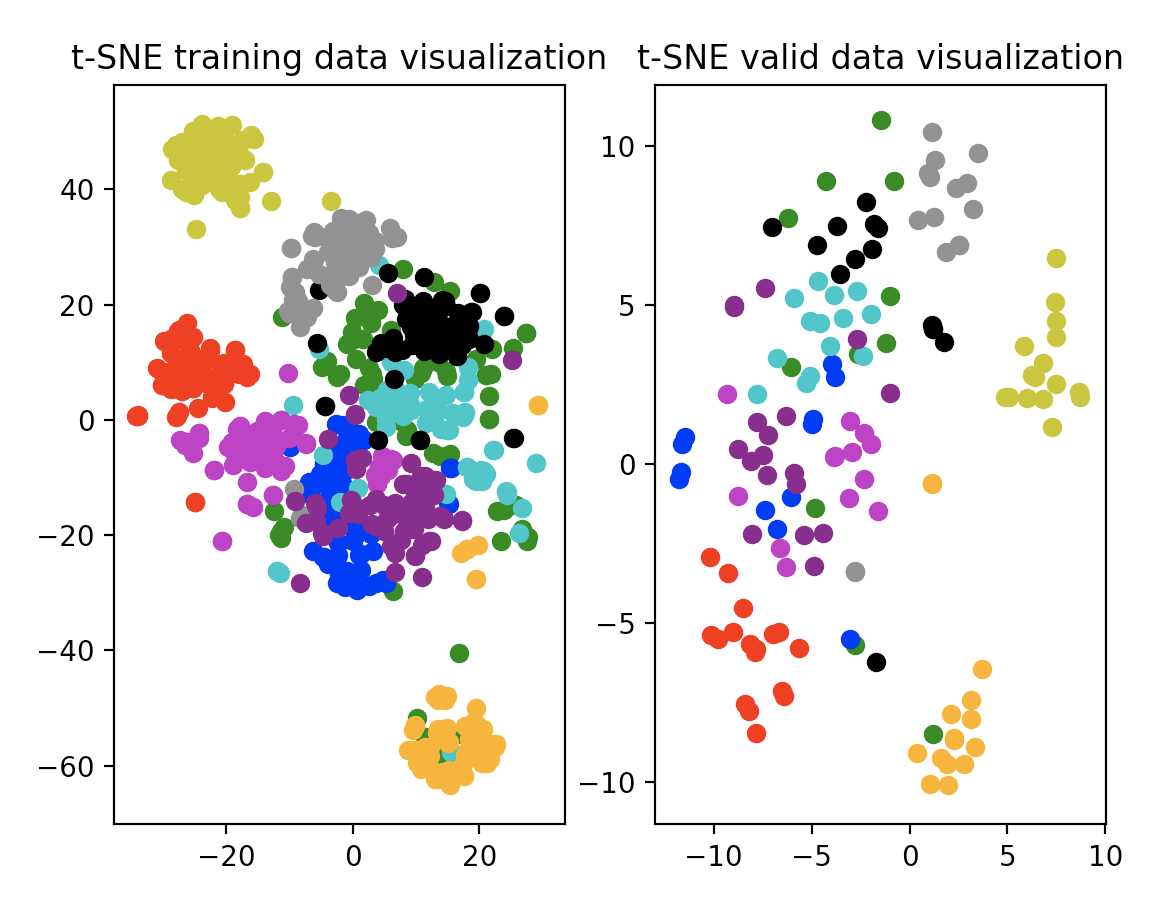
The training loss is approximately equal to validation loss, so I think it’s not underfitting nor overfitting.

1. Pick the first 10 identities from dataset. Visualize the features of training/validation data you extract from pretrained alexnet and your own model to 2D space with t-distributed stochastic neighbor embedding (t-SNE), compare the results and explain the difference. (5%)

ALEXNET:



MY MODEL:



The graphs above are visualization of feature extractions of Alexnet and my model perspectively. Alexnet extracts from “features” layer, and my model extracts from “layer3”.

We can see that the result of t-SNE of Resnet is better. I think it is mainly because the Resnet uses more convolution layer to get features of images, so when it projects data with t-SNE, the clustering are split more obviously.