CS698: Assignment 5

Ronghao Yang
ID: 20511920
Session: 4:00-5:20pm

November 30, 2017

1 Exercise 1

1.1 Exercise 1.1

The VGG11 net has been modified for this assignment. The reason is that the original VGG11 net is too complex, our dataset may not need such a complex model. A simplified model may be enough for MNIST dataset. The modified architecture used is the following:

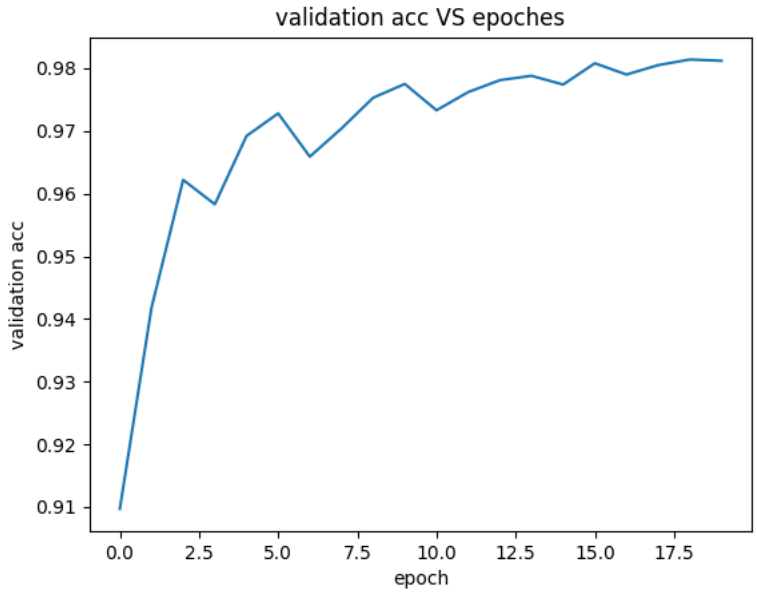
My CNN Net
Conv3-64
maxpooling
Conv3-128
maxpooling
Conv3-256
maxpooling
FC-512
FC-10
soft-max

The parameter used are:
batch_size = 64
learning_rate = 0.01
momentum = 0.9
epochs = 20
In the original question, we are asked why we need to resize the image from 28×28 to 32×32, this reason is the following:
In VGG11, there exist 5 maxpooling layers, after each maxpooling layer, the size of the image is reduced by 2 (eg from 32×32 to 16×16), with the image size being 28×28, we are unable to do 5 times of maxpooling.

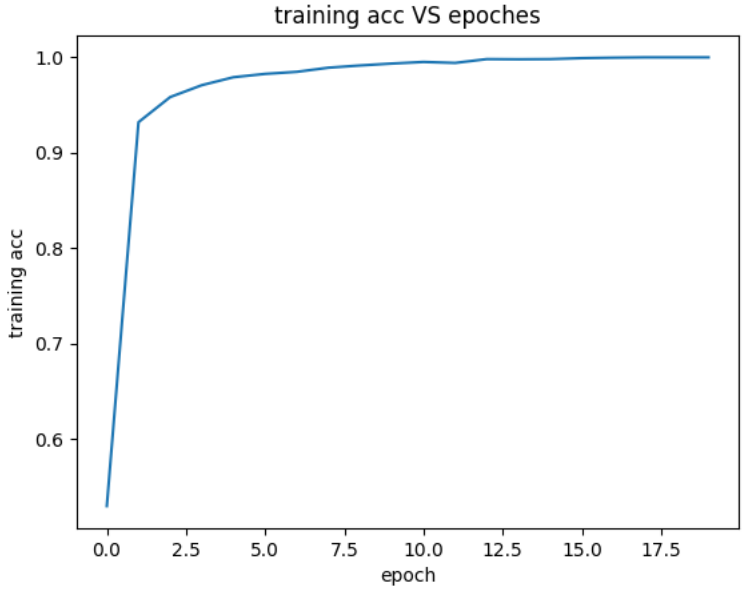
1.2 Exercise 1.2

For this section, the number of epochs is set to be 10.

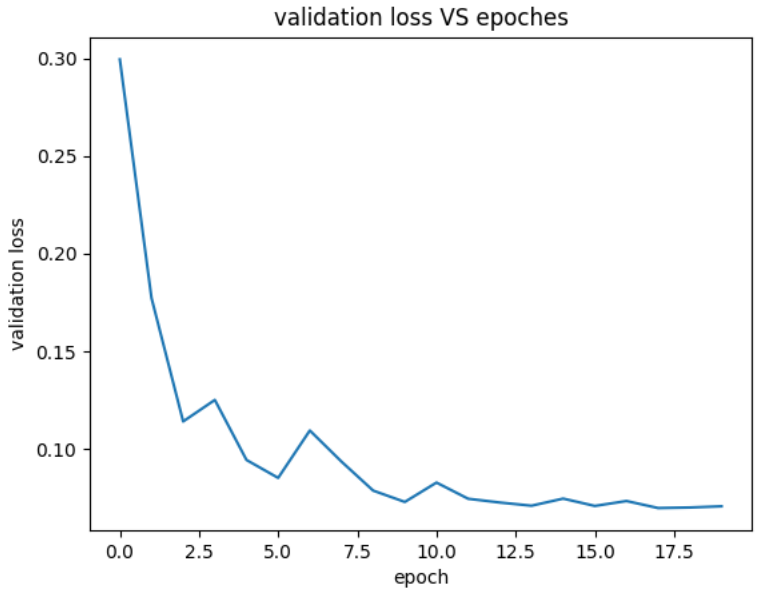
•test accuracy vs the number of iterations



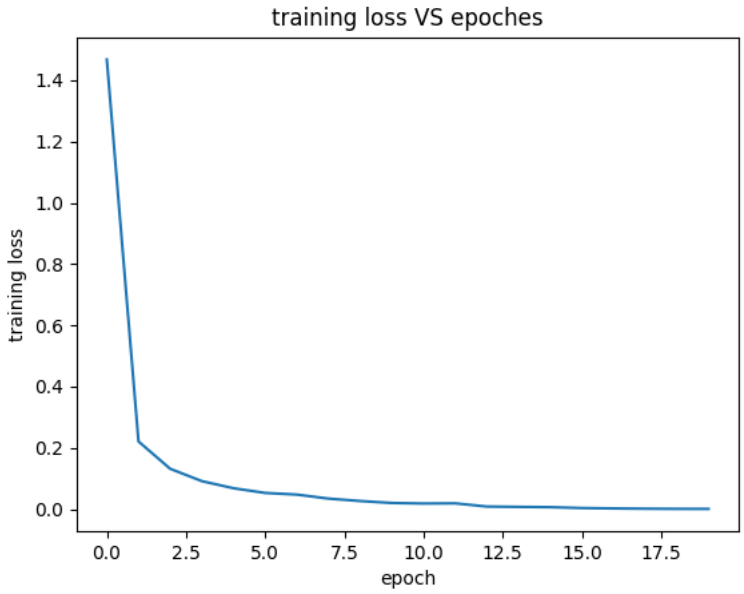
•training accuracy vs the number of iterations



•test loss vs the number of iterations



•training loss vs the number of iterations

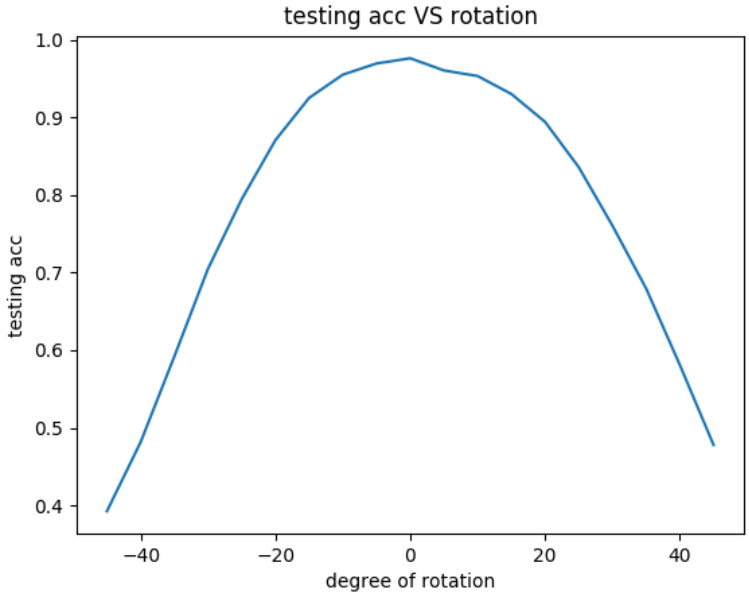


1.3 Exercise 1.3

For this section, the number of epochs is set to be 10.

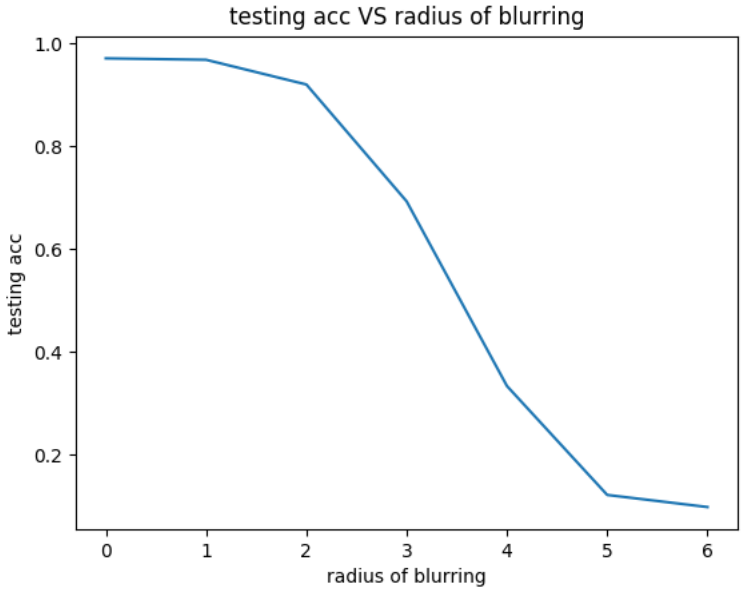
•test accuracy vs the degree of rotation

The more degrees of rotation we apply on the testing set, the lower the testing accuracy is.



•test accuracy vs Gaussian blurring

The higher radius of Gaussian radius we apply on the testing set, the lower the testing accuracy is.

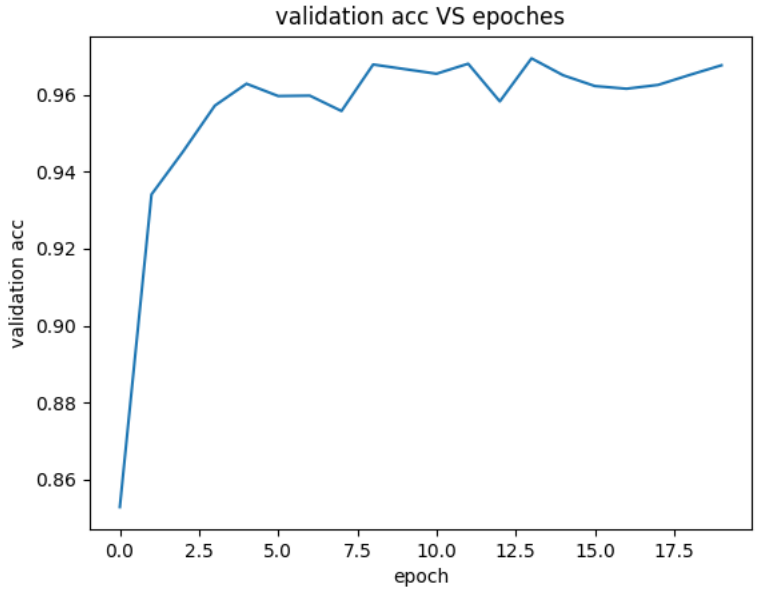


1.4 Exercise 1.4

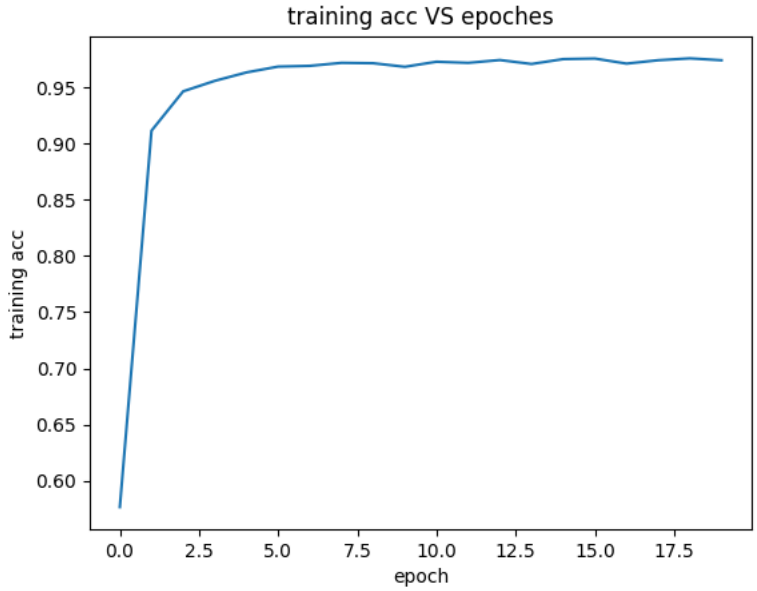
•l2 regularization

As we can see here, with l2 regularization, the overall performance has not been affected much. The accuracy with regularization is slightly lower than the accuracy without regularization, this may comes from the complexity of the model. Our simplified model only has a few layers, adding regularization terms to it makes our model even more simplified.

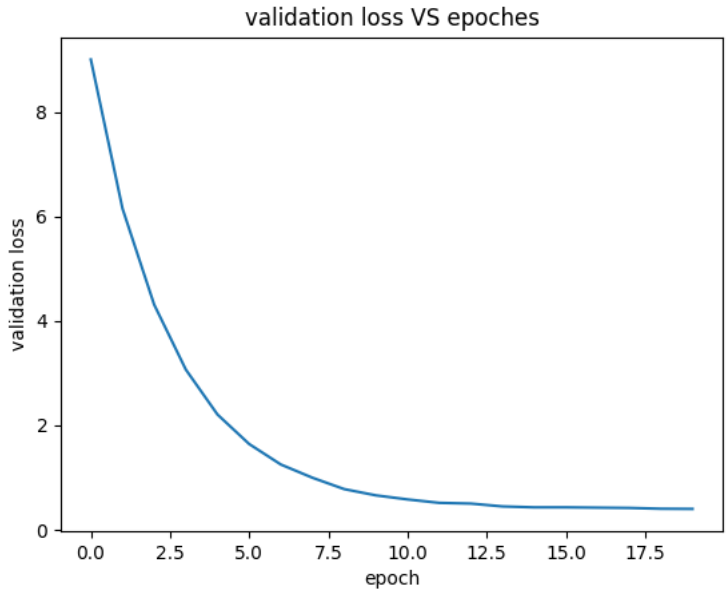
•test accuracy vs the number of iterations



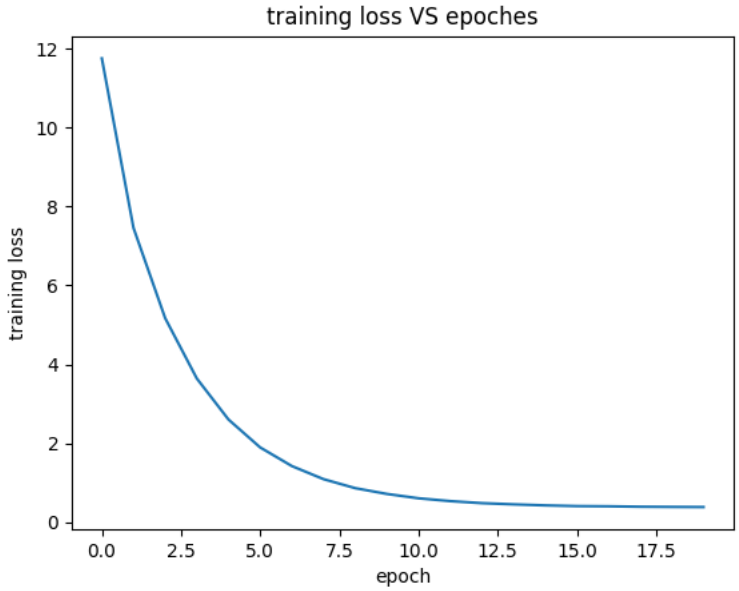
•training accuracy vs the number of iterations



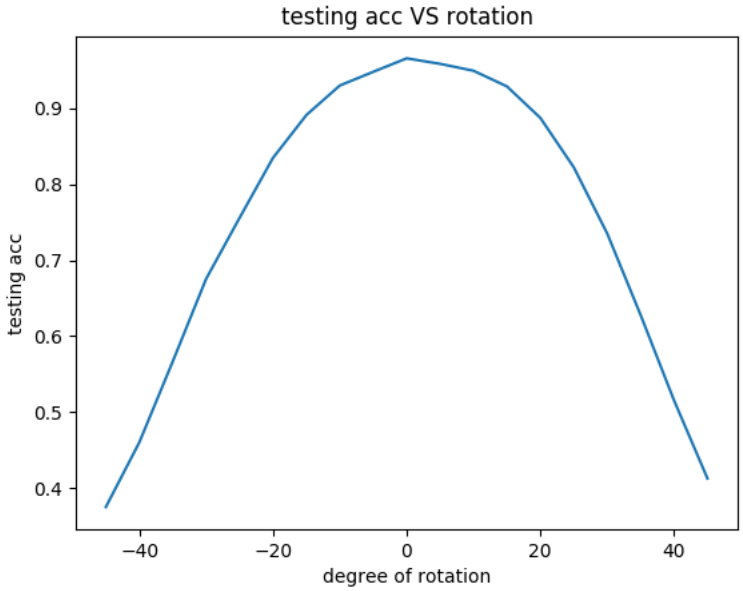
•test loss vs the number of iterations



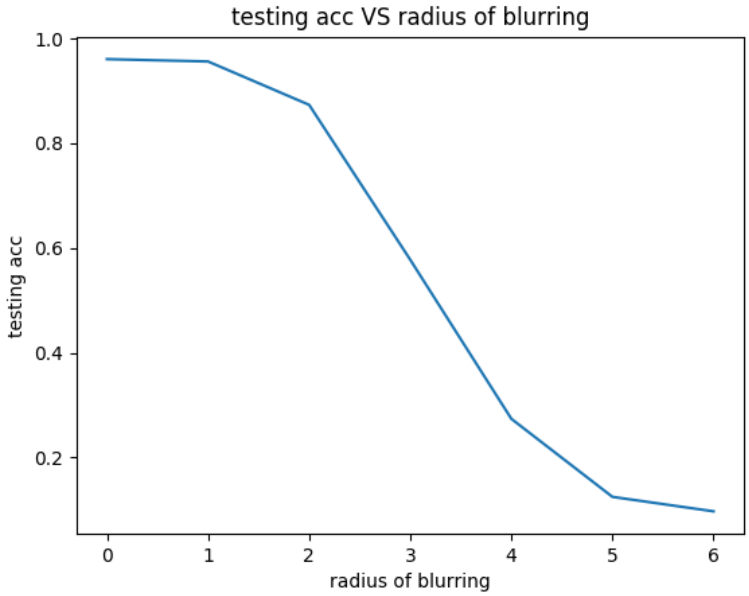
•training loss vs the number of iterations



•test accuracy vs the degree of rotation



•test accuracy vs Gaussian blurring



•Data augmentation

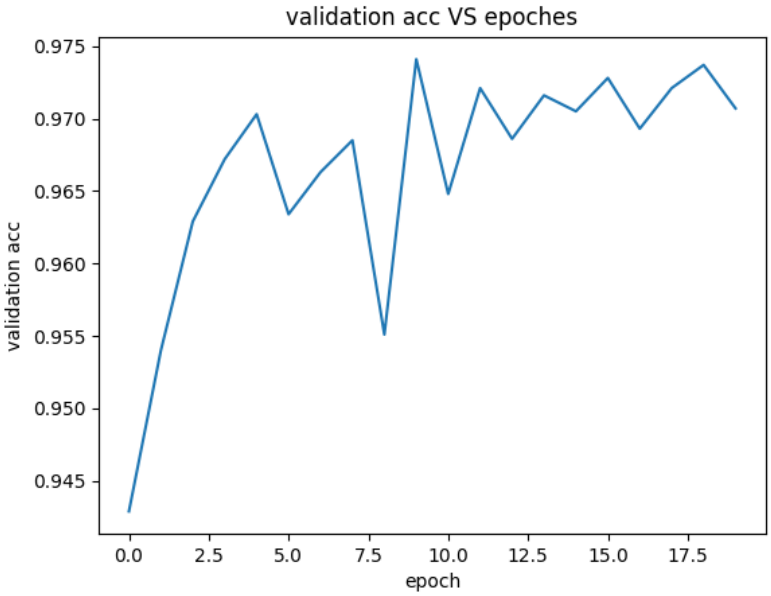
The original training set has 6000 images, for data augmentation, I have added another 6000*7 additional modified images to it. For those extra 7 batches, the details of the data augmentation are:

Batch 1: image rotation -45 degrees
Batch 2: image rotation 45 degrees
Batch 3: image rotation -25 degrees
Batch 4 image rotation 25 degrees
Batch 5: image with Gaussian Blur radius = 2
Batch 6: image with Gaussian Blur radius = 4
Batch 7: image with Gaussian Blur radius = 6

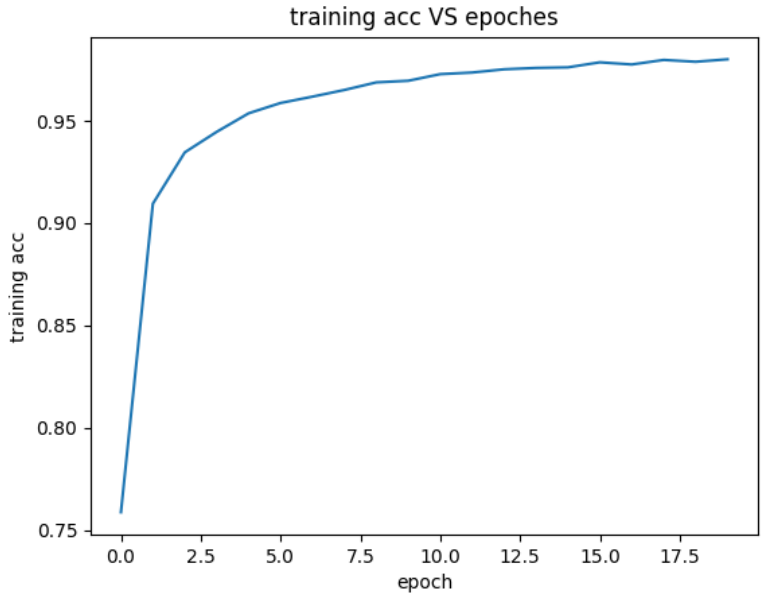
With data augmentation, we can see that on the datasets with rotation of Gaussian blurring, the accuracy has significantly increased.

In the plots of validation loss and validation accuracy, there exist some huge fluctuations, there might exist some over-fitting problems.

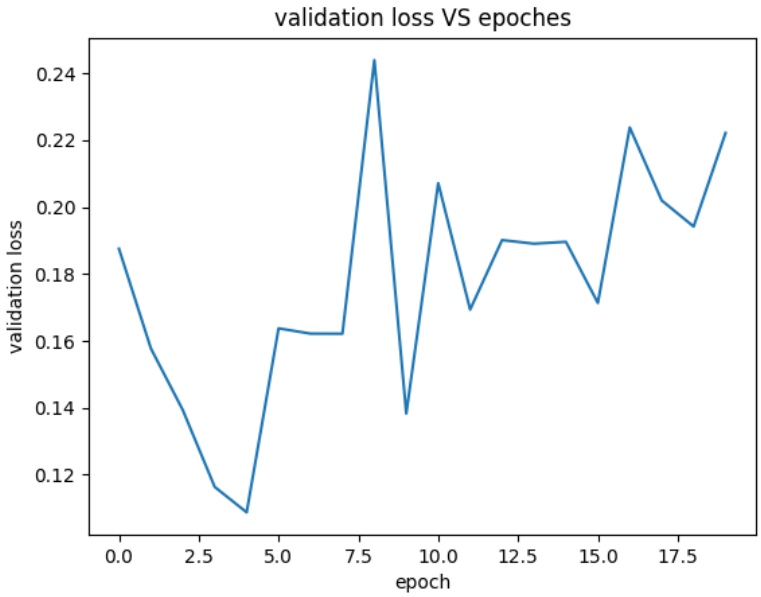
•test accuracy vs the number of iterations



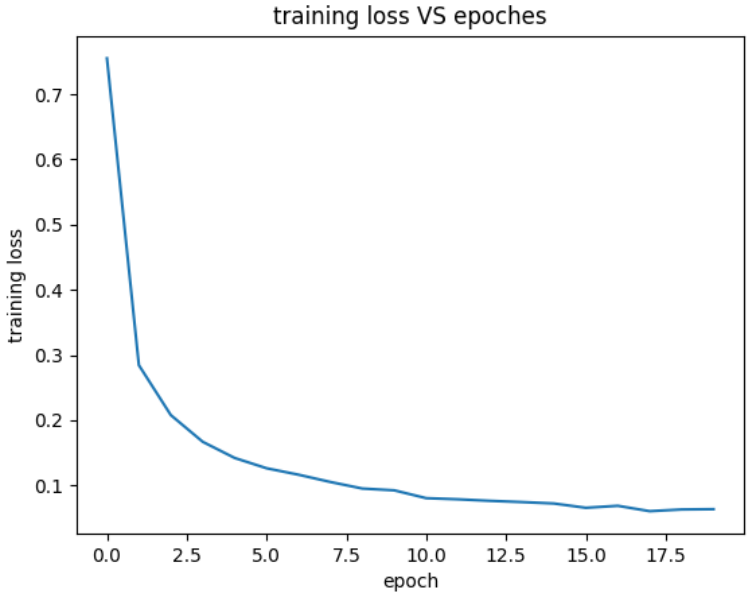
•training accuracy vs the number of iterations



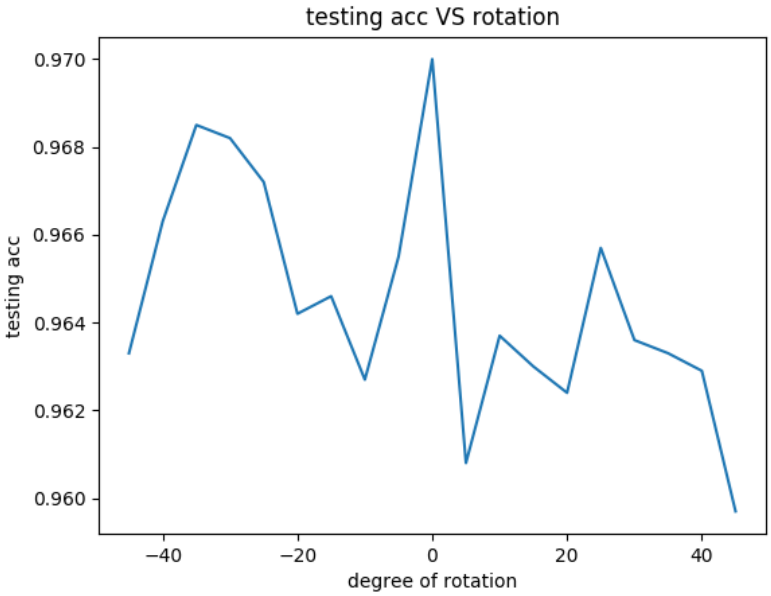
•test loss vs the number of iterations



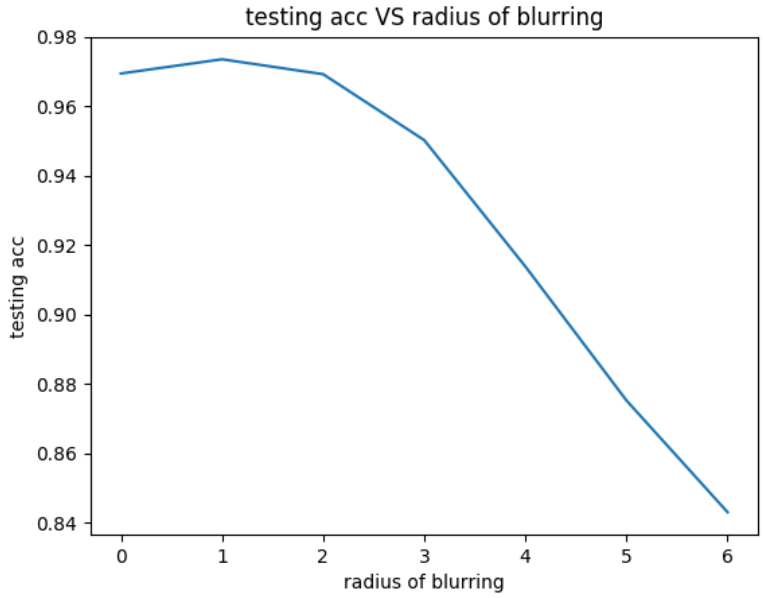
•training loss vs the number of iterations



•test accuracy vs the degree of rotation



•test accuracy vs Gaussian blurring



2 Exercise 2

• When fixing a feature j, we only need to consider all the values of X_{ij}s, since any other thresholds don't make a difference in clustering X into 2 groups. Therefore, only n value are needed to be considered for t_js.

• Sorting all X_{ij}s takes $nlog(n)$ time.

• When computing μ and ν, we set the derivatives of both $\sum_{i:X_{ij}\leq t_j}(y_i-\mu)^2$ and $\sum_{i:X_{ij}>t_j}(y_i-\nu)^2$ to be 0, and calculate the optimal values for μ and ν. The optimal values are $\mu=\frac{\sum_{i:X_{ij}\leq t_j} y_i}{n_1}$, $\nu=\frac{\sum_{i:X_{ij}>t_j} y_i}{n_2}$, where n_1 is the number of y_is when $X_{ij}\leq t_j$, n_2 is the number of y_is when $X_{ij}>t_j$.

• There are d different features, we need to loop d times.

• Algorithm that runs in $O(d\ n\ logn)$:

Data: $X\in R^{n\times d}$, $y\in R^n$
Result: $Jjbest$, t_{best}
$Loss_j=0$ for all j
$T_j=0$ for all j
for *Feature j* ***in*** *1...d* **do**
  $loss_i=0$ for all i
  Sort $X_{:j}$, so that $X_{1j}\leq X_{2j}\leq ....\leq X_{nj}$
  Let $S_1=0$, $S_2=\sum_{k=1}^{i} y_i$
  Let $S_3=\sum_{k=1}^{i} y_i^2$
  for *Threshold* t_j ***in*** X_{1j}, X_{2j}, ..., X_{nj} **do**
    $S_1=S_1+y_i$, $S_2=S_2-y_i$ when $t_j=X_{ij}$;
    $loss_i=S_3-\frac{S_1^2}{i}-\frac{S_2^2}{n-i}$
  end
  $Loss_j=\min loss$, for all i
  $T_j=argmin_{t_j}\ loss$
end
$j_{best}=argmin_j\ Loss$
$t_{best}=T_{J_{best}}$

Algorithm 1: Regression Tree Algorithm