CSC 311 Assignment 2

Peiqing Yu 1003769004

1.

(a). Since in data space, the positive and negative region are both convex, if we draw the line segment between two positive example x=3 and x=-1, then this entire line segment must be classified as positive. But the negative example x=1 lied on this line, which means this point must be classified as both positive and negative, which is impossible. Therefore, this dataset is not linearly separable.

(b). In this feature representation, our training set becomes:

x	$\varphi_1(x)$	$\varphi_2(x)$	t
-1	-1	1	1
1	1	1	0
3	3	9	1

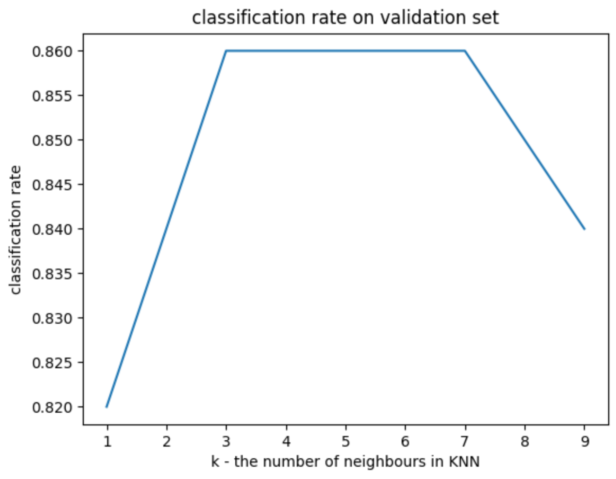
We can then write out the inequalities corresponding to each training case:

$$w_1(-1) + w_2 \geq 0$$
$$w_1 + w_2 < 0$$
$$3w_1 + 9w_2 \leq 0$$

And we can let (w_1, w_2) = (-2, 1).

2.1

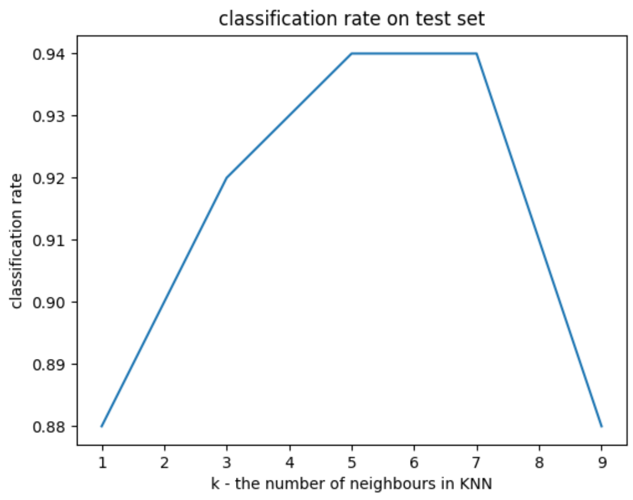
(a). Please check the code in run_knn.py, below is the plot of classification rate on the validation set of the corresponding k.



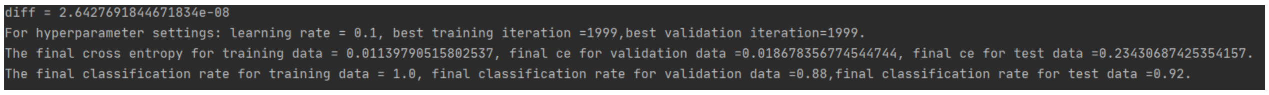
(b). The best performance of the classifier of validation set is at k = 3, 5, 7. Even not for best performance k, this classifier still has a general high accuracy which is above 0.8.

I would choose k^* = 5 to set my hyperparameter, since it has the highest classification rate for

validation set which is 0.86.
For k = 3, k= 7, they have the same validation accuracy 0.86 (see Fig1 in appendix).
The test performance of these k values matches the validation performance, with k =3, 5, 7 have the highest accuracy.

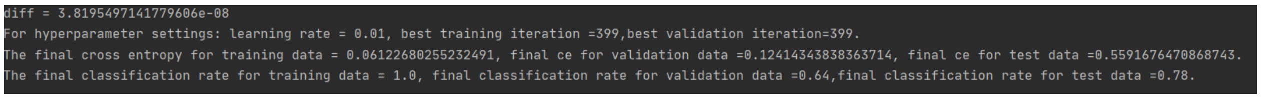


2.2
(a). Please check logistic.py
(b). best hyperparameter setting for mnist_train:
Learning_rate = 0.1, num_iterations = 2000, weights = **0**
Final cross entropy and classification error on the training, validation and test sets:



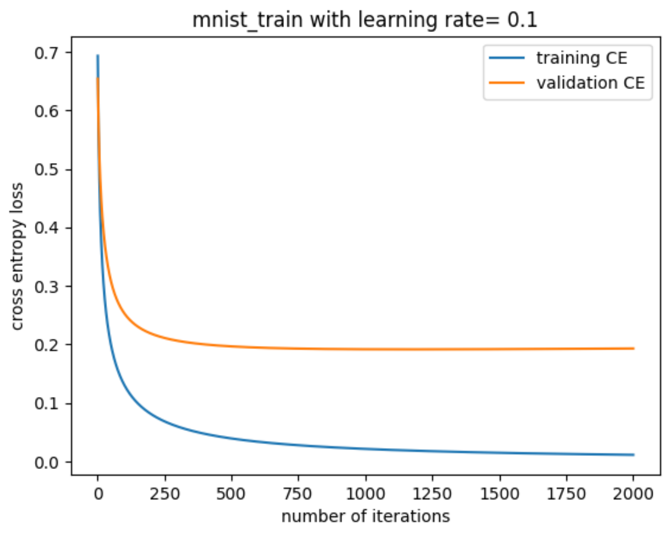
```
diff = 2.6427691844671834e-08
For hyperparameter settings: learning rate = 0.1, best training iteration =1999,best validation iteration=1999.
The final cross entropy for training data = 0.01139790515802537, final ce for validation data =0.018678356774544744, final ce for test data =0.23430687425354157.
The final classification rate for training data = 1.0, final classification rate for validation data =0.88,final classification rate for test data =0.92.
```

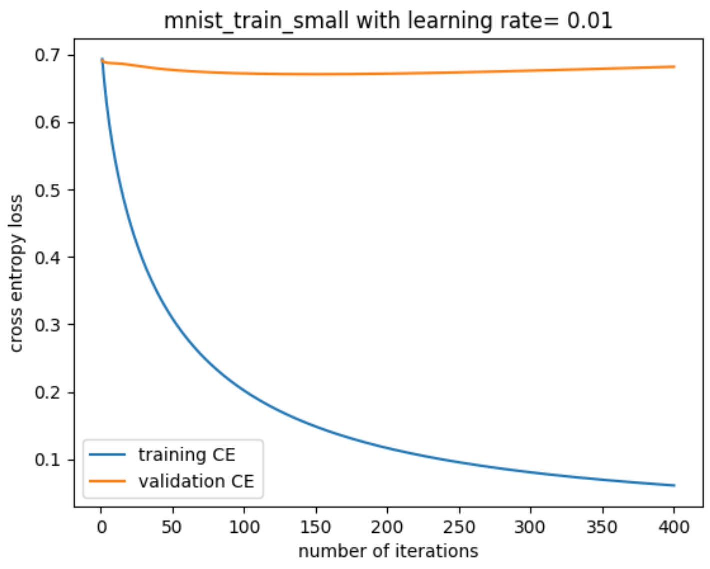
Best hyperparameter setting for mnist_train_small:
Learning_rate = 0.01, num_iterations = 400, weights = **0**
Final cross entropy and classification error on the training, validation and test sets:



```
diff = 3.8195497141779606e-08
For hyperparameter settings: learning rate = 0.01, best training iteration =399,best validation iteration=399.
The final cross entropy for training data = 0.06122680255232491, final ce for validation data =0.12414343838363714, final ce for test data =0.5591676470868743.
The final classification rate for training data = 1.0, final classification rate for validation data =0.64,final classification rate for test data =0.78.
```

(c). As the training progresses, we can check the cross entropy of mnist_train and mnist_train_small.
There's not much change of the results

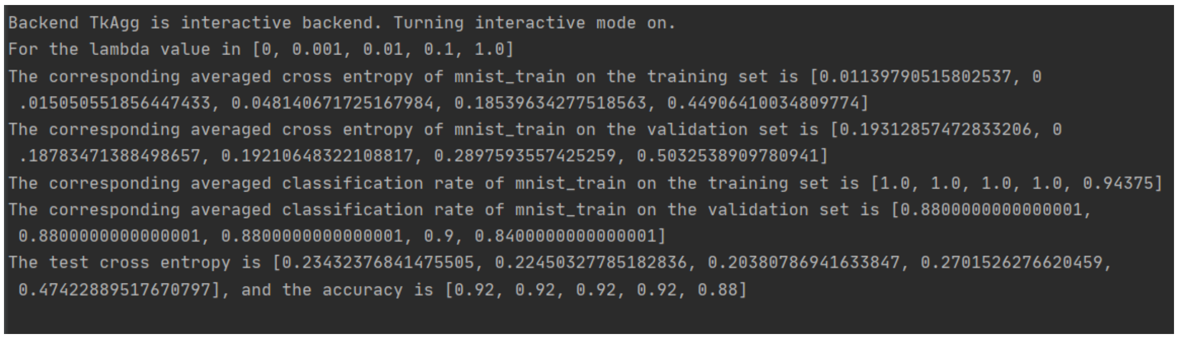




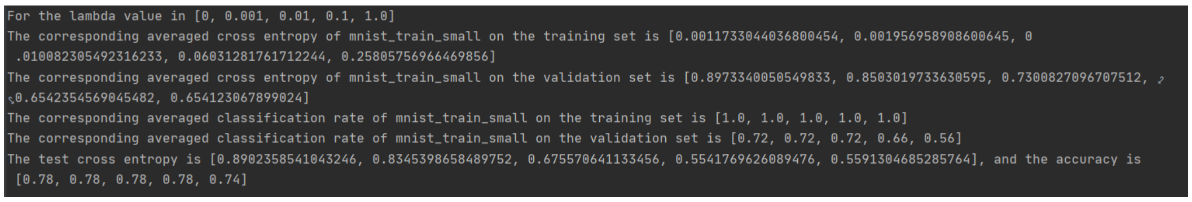
2.3

(a). Please check logistic.py

(b). The averaged cross entropy and classification error on the training and validation sets for each lambda are:

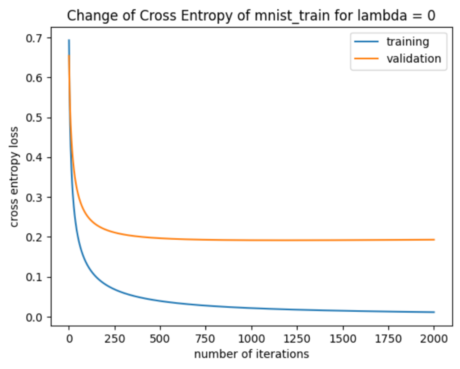


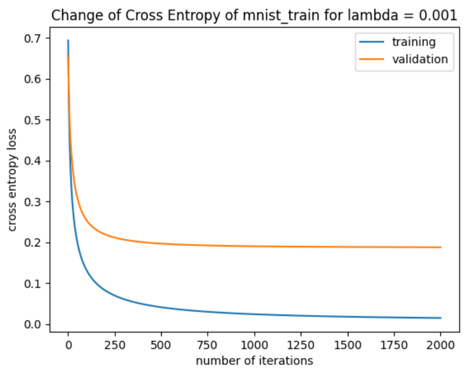
```
Backend TkAgg is interactive backend. Turning interactive mode on.
For the lambda value in [0, 0.001, 0.01, 0.1, 1.0]
The corresponding averaged cross entropy of mnist_train on the training set is [0.01139790515802537, 0
 .015050551856447433, 0.048140671725167984, 0.18539634277518563, 0.44906410034809774]
The corresponding averaged cross entropy of mnist_train on the validation set is [0.19312857472833206, 0
 .18783471388498657, 0.19210648322108817, 0.2897593557425259, 0.5032538909780941]
The corresponding averaged classification rate of mnist_train on the training set is [1.0, 1.0, 1.0, 1.0, 0.94375]
The corresponding averaged classification rate of mnist_train on the validation set is [0.8800000000000001,
 0.8800000000000001, 0.8800000000000001, 0.9, 0.8400000000000001]
The test cross entropy is [0.23432376841475505, 0.22450327785182836, 0.20380786941633847, 0.2701526276620459,
 0.47422889517670797], and the accuracy is [0.92, 0.92, 0.92, 0.92, 0.88]
```

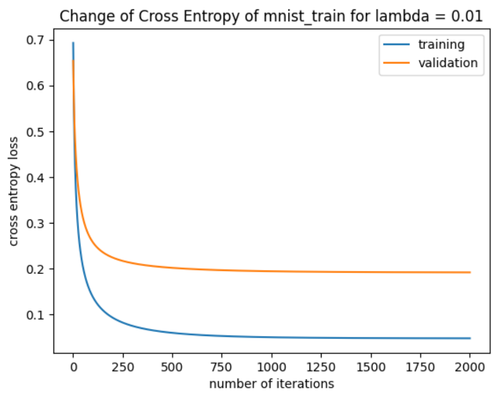


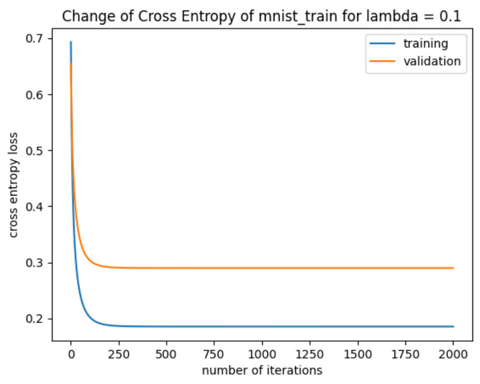
```
For the lambda value in [0, 0.001, 0.01, 0.1, 1.0]
The corresponding averaged cross entropy of mnist_train_small on the training set is [0.0011733044036800454, 0.001956958908600645, 0
 .010082305492316233, 0.06031281761712244, 0.25805756966469856]
The corresponding averaged cross entropy of mnist_train_small on the validation set is [0.8973340050549833, 0.8503019733630595, 0.7300827096707512, 
0.6542354569045482, 0.654123067899024]
The corresponding averaged classification rate of mnist_train_small on the training set is [1.0, 1.0, 1.0, 1.0, 1.0]
The corresponding averaged classification rate of mnist_train_small on the validation set is [0.72, 0.72, 0.72, 0.66, 0.56]
The test cross entropy is [0.8902358541043246, 0.8345398658489752, 0.675570641133456, 0.5541769626089476, 0.5591304685285764], and the accuracy is
 [0.78, 0.78, 0.78, 0.78, 0.74]
```

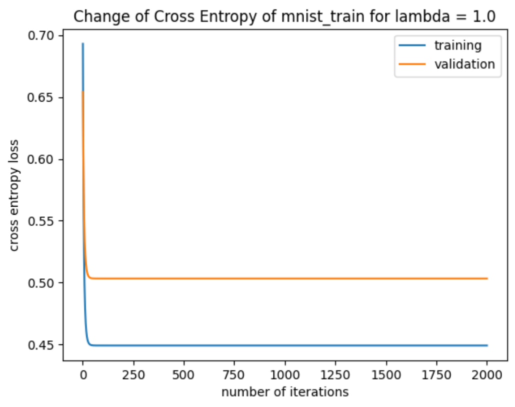
For mnist_train, the cross-entropy changes as training progresses for each lambda are:



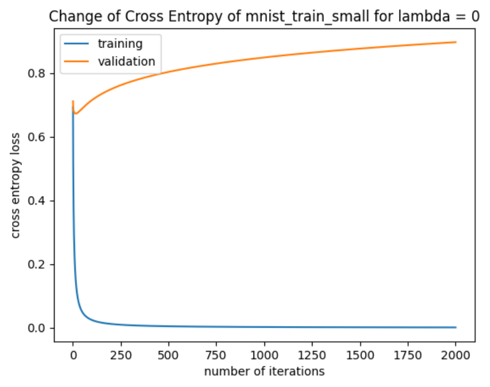


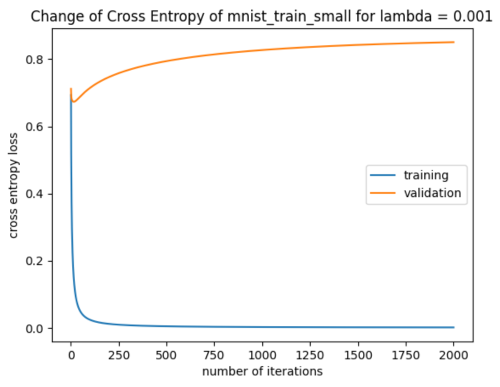


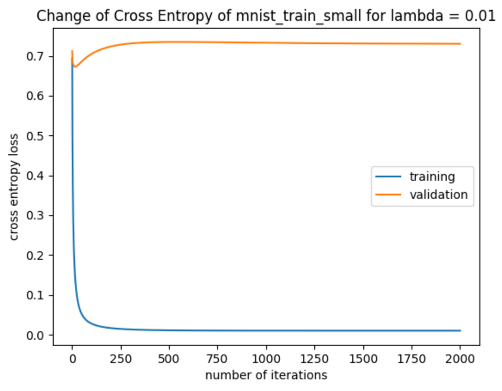


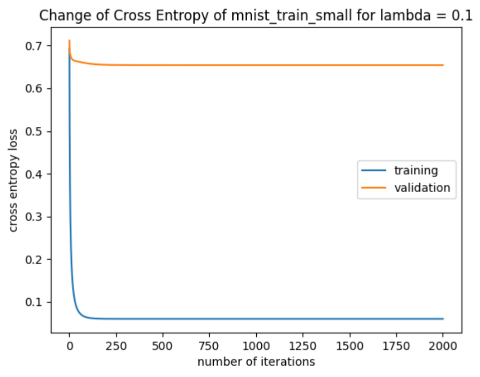


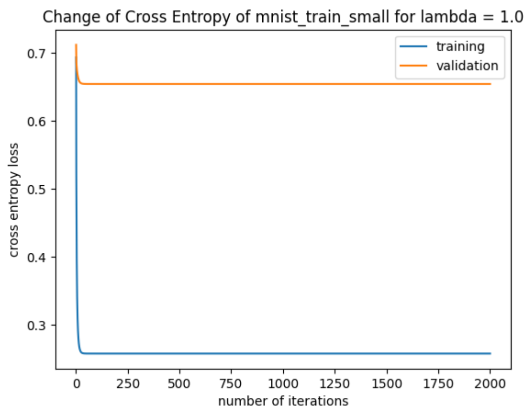
For mnist_train_small, the cross-entropy changes as training progresses for each lambda are:











(c). For mnist_train, as we increase lambda, the cross entropy of train goes up, the cross entropy of validation first goes down then goes up; for mnist_train_small, the cross entropy of train and validation nearly stays the same.

The penalty rate lambda adds squared magnitude of coefficient as penalty term to the loss function, if lambda is too large then the coefficient will be restricted to bebigger and thus model might be underfitting. However, as we see from lambda = 0, small penalty rate may cause the algorithm to converge slowly. For the mnist_train dataset, we see a first down then up loss curve for validation set and a general increasing loss in training set, this might indicate the choice of best lambda lies at the lowest point at validation dataset. For a smaller dataset, we can see there is not such a large change in cross entropy

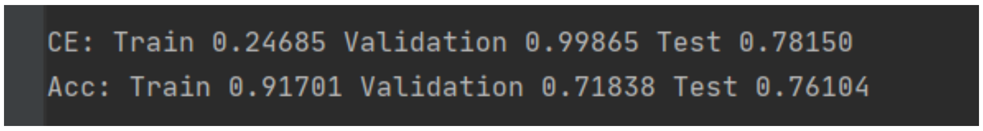
For mnist_train, the best lambda = 0.001, the test ce = 0.2245 and acc = 0.92

For mnist_train_small, the best lambda = 0.01, the test ce = 0.6756 and acc =0.78.

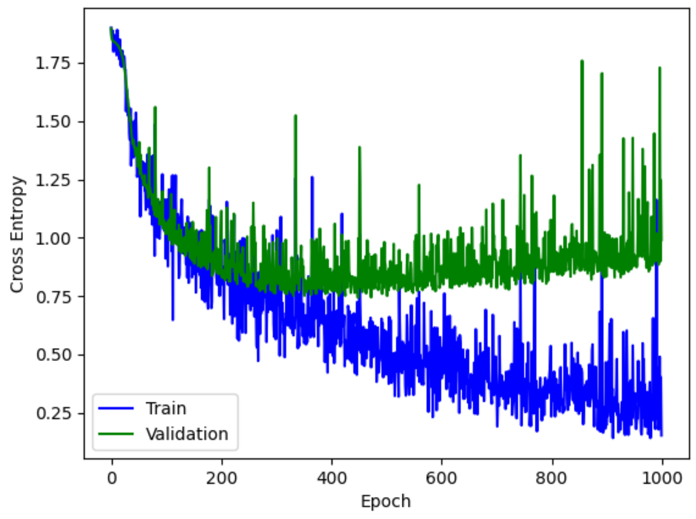
3.

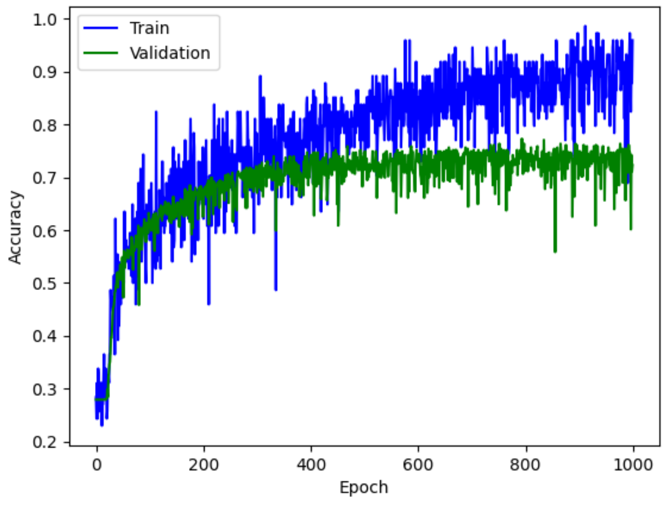
(a). Please check nn.py

(b). The training, validation and test error of default hyperparameters are:



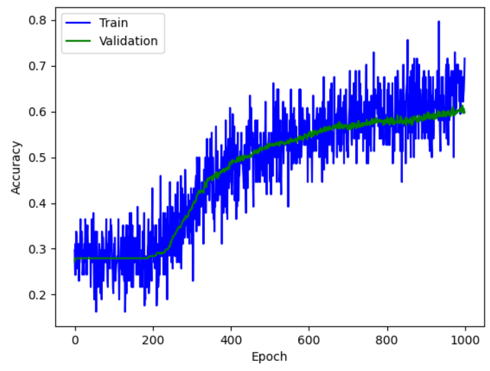
```
CE: Train 0.24685 Validation 0.99865 Test 0.78150
Acc: Train 0.91701 Validation 0.71838 Test 0.76104
```

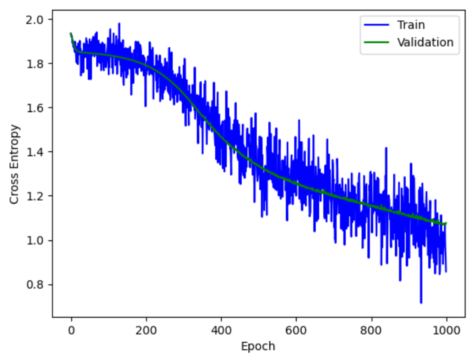


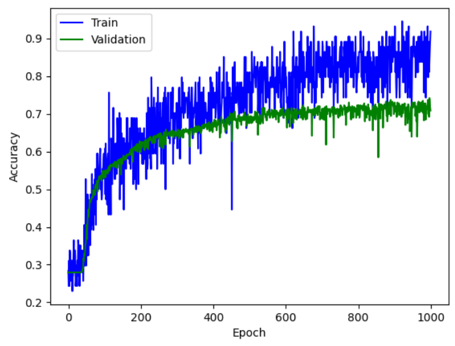


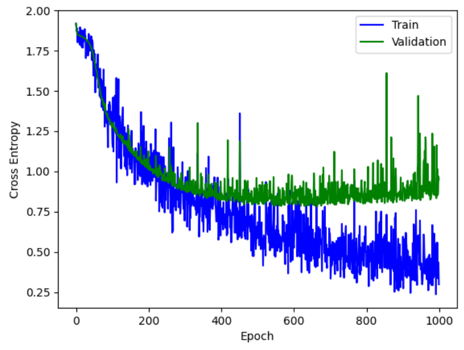
For cross entropy plot, the validation set first decrease then goes up as learning progresses; however, the training set decreases all the way as we always decrease the gradient at back propagation and converges at the end. This difference may due to possible overfitting. For the accuracy, we can see both validation and training increase as epoch increases. However, the validation first converges, while the training still increases until we hit the maximum accuracy 1.

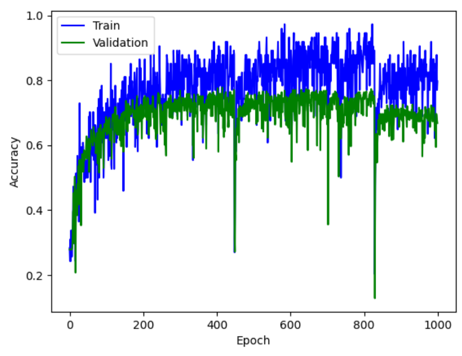
(c). Now hold the other parameters, and try learning rate of 0.001, 0.005, 0.05, 0.1, 0.5

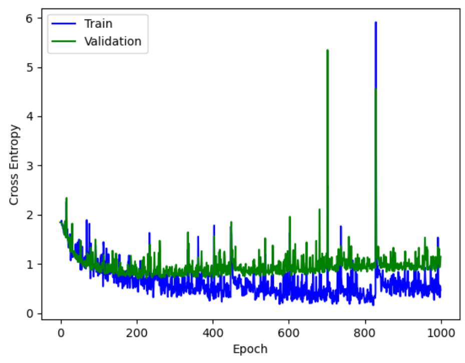

Train
Validation
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
Epoch

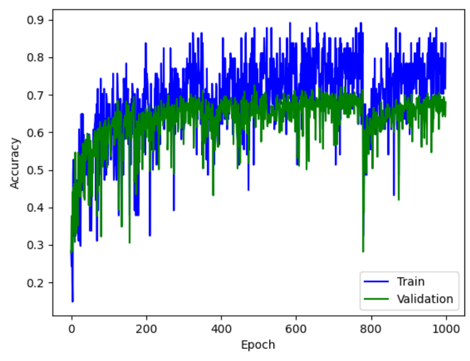

Train
Validation
Cross Entropy
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0
200
400
600
800
1000
Epoch

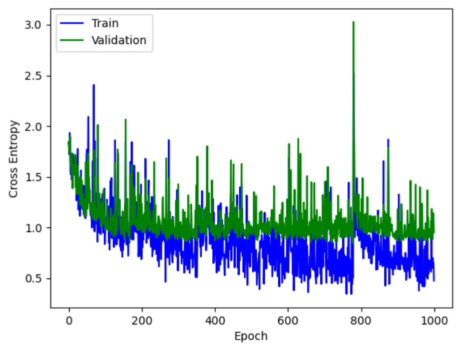

Train
Validation
Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
Epoch

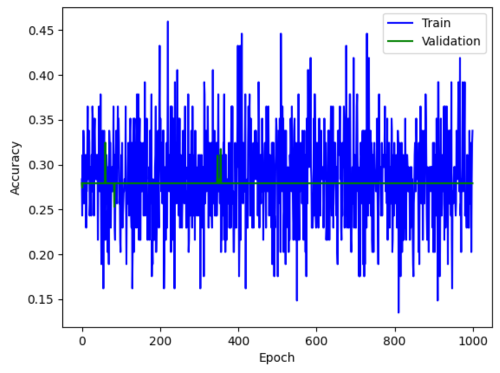

Train
Validation
Cross Entropy
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
200
400
600
800
1000
Epoch

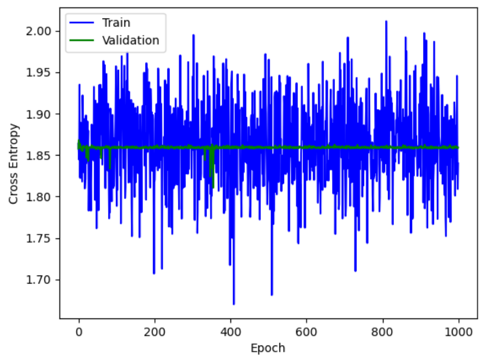

Train
Validation
Accuracy
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000
Epoch


Train
Validation
Cross Entropy
6
5
4
3
2
1
0
0
200
400
600
800
1000
Epoch


Train
Validation
Accuracy
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
200
400
600
800
1000
Epoch

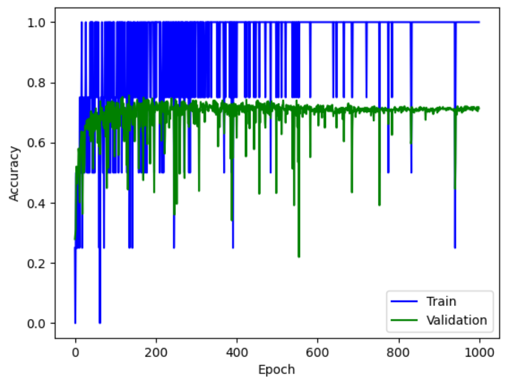

Train
Validation
Cross Entropy
3.0
2.5
2.0
1.5
1.0
0.5
0
200
400
600
800
1000
Epoch

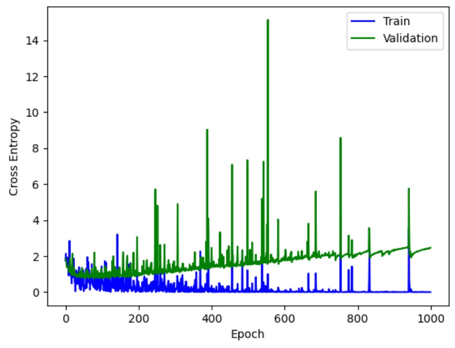


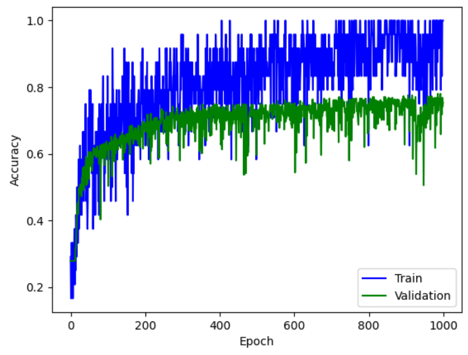


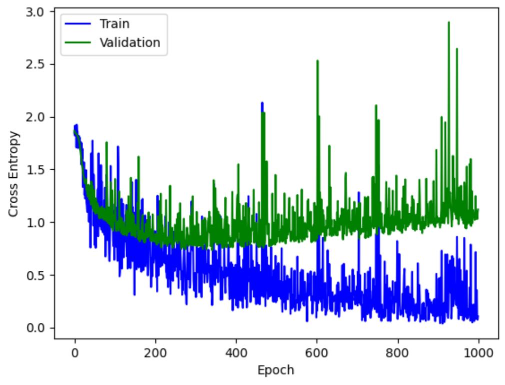
The learning rate is a hyperparameter that controls how much the gradient is updates as the steepest direction. Smaller learning rates require more training epochs because they give smaller changes to the weights each update; however, larger learning rates result in rapid changes and require fewer training epochs but at a cost of more noise or oscillation. This leads to that the larger learning rate converges quickly i.e. learning rate = 0.5. Also note that, for either too small or too large learning rate, the validation accuracy and cross entropy seems to have little variance.
For our model, I would choose learning rate = 0.05, which converges not bad while have less noise.

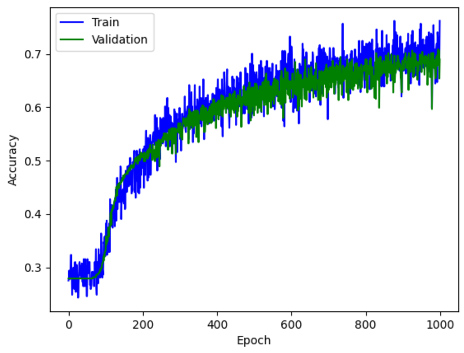
Now hold the other parameters, and try mini_batch size of 10, 50, 500, 700, 1000

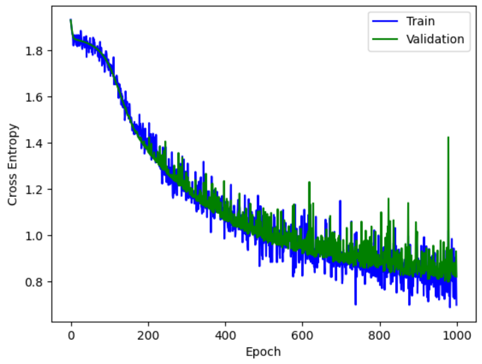


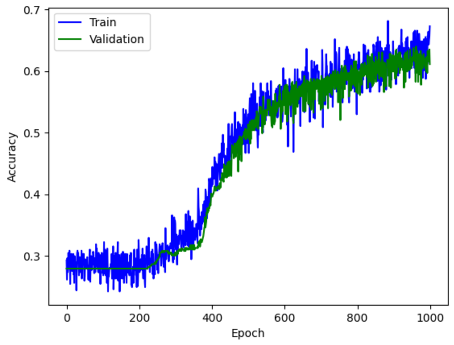


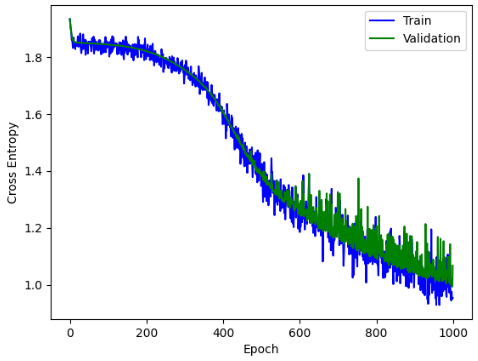


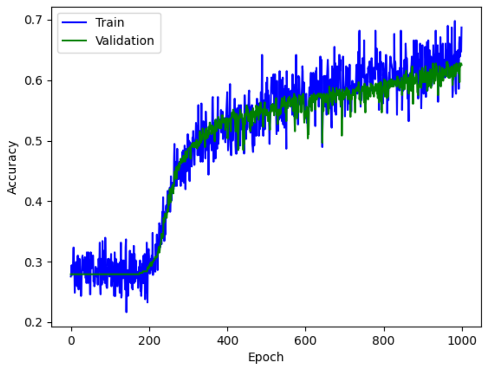


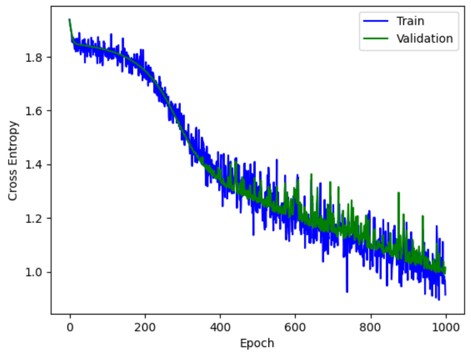








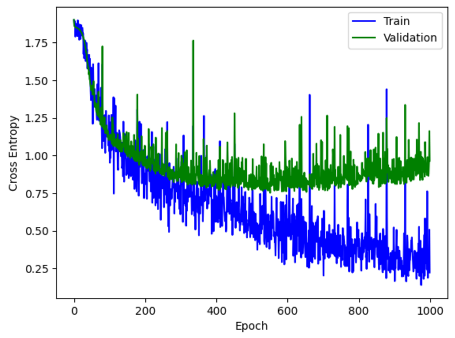


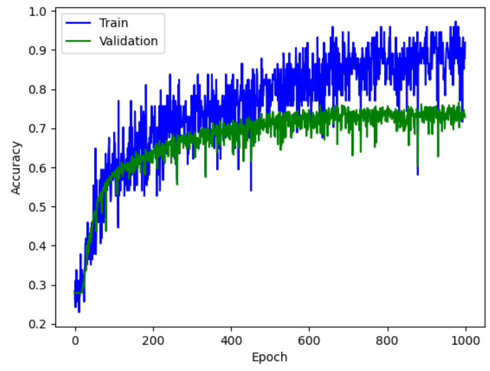


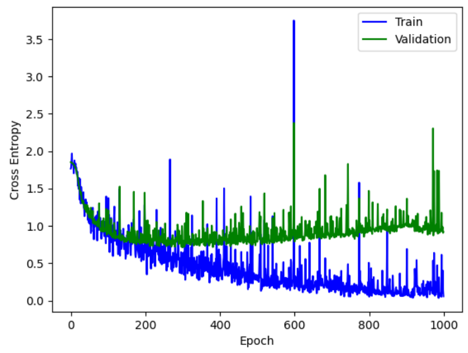
The number of examples from the training dataset used in the estimate of the error gradient is called the batch size. We can see that small value of batch size converges quickly for both cross entropy and accuracy, but at a cost with high variance or noise. However, as we increase the number of batch size, we see that the algorithm converges slowly but with less noise and thus the validation closer to the training set. This means we may need more computations and memories.

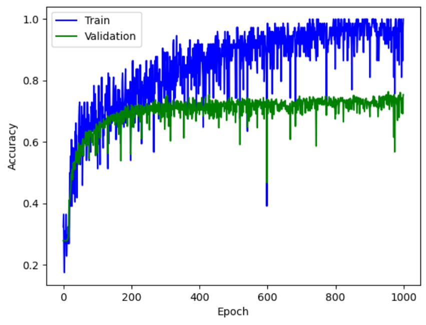
For our model, I would choose the batch size between 50-500, maybe at 200 to balance the convergence time and cost of noise.

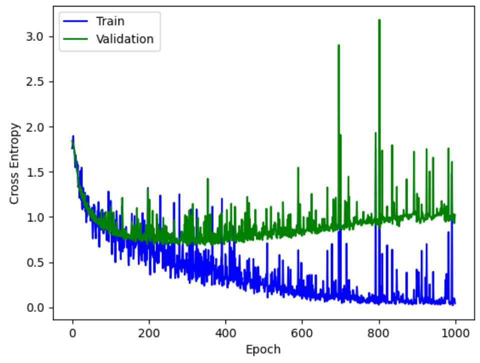
(d). Now try 3 different combination of hidden units number: (16,16), (80,32), (80,80)

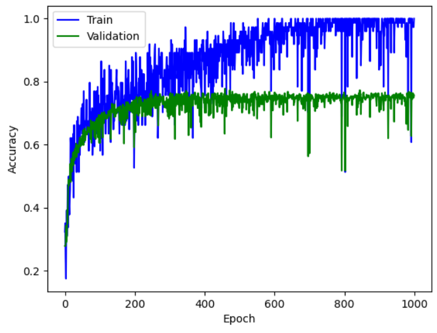










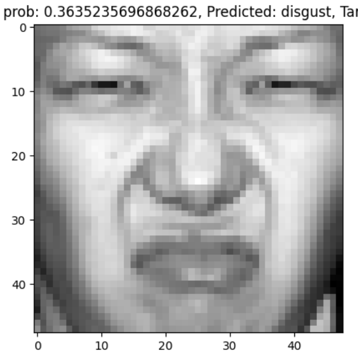


We can see that as the number of hidden units increases, the neural network algorithm is better at memorization and bad at generalization. We can see that for (80, 80), the train set nearly hit 100% quickly (converges quickly), however it not generalizes the validation dataset. We should pick a middle range hidden units combination to prevent over generalization.

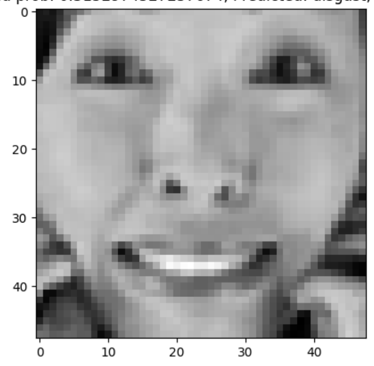
(e). Randomly present 8 examples where the neural network is not confident of the classification output.

The pictures that I chose are having the highest predictions lower than threshold 0.5. However, when we compare the prediction and the result, most of the unconfident predictions seem to be wrong. These pictures with low prediction probabilities tight to be hard to classify even for human, however our neural network model may need to be modified to improve prediction.

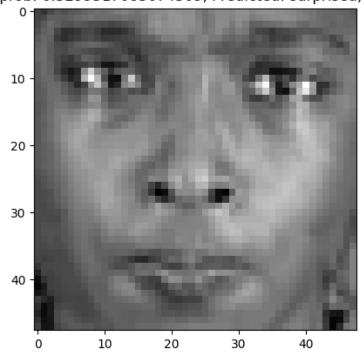
Predicted prob: 0.3635235696868262, Predicted: disgust, Target: disgust



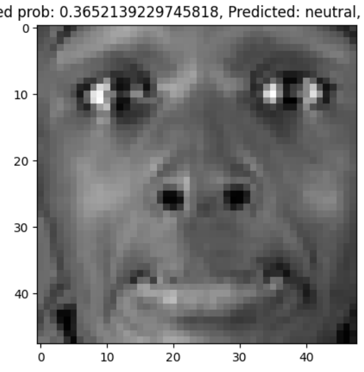
Predicted prob: 0.3152974527257674, Predicted: disgust, Target: fear



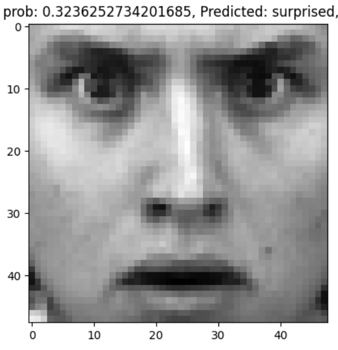
Predicted prob: 0.3299317083674309, Predicted: surprised, Target: anger



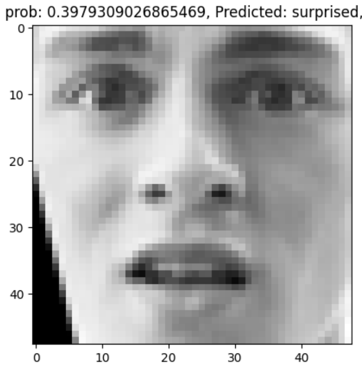
Predicted prob: 0.3652139229745818, Predicted: neutral, Target: fear



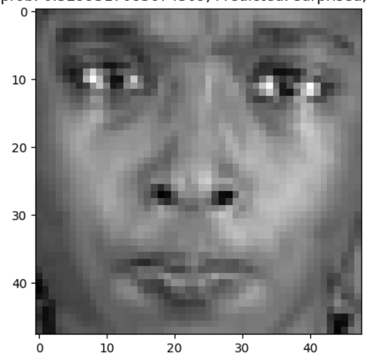
Predicted prob: 0.3236252734201685, Predicted: surprised, Target: fear



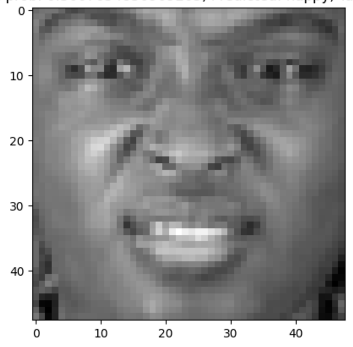
Predicted prob: 0.3979309026865469, Predicted: surprised, Target: fear



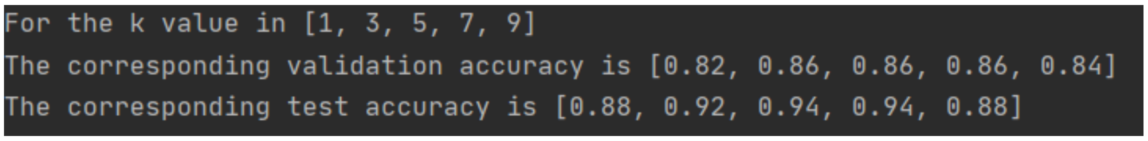
Predicted prob: 0.3299317083674309, Predicted: surprised, Target: anger



Predicted prob: 0.3007654850909205, Predicted: happy, Target: disgust

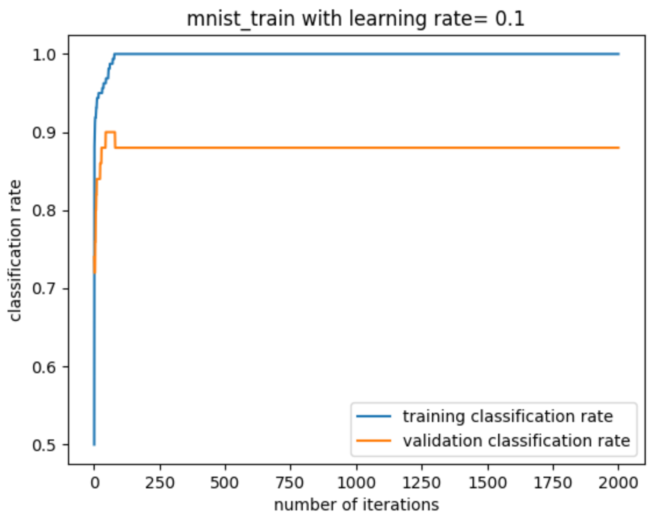


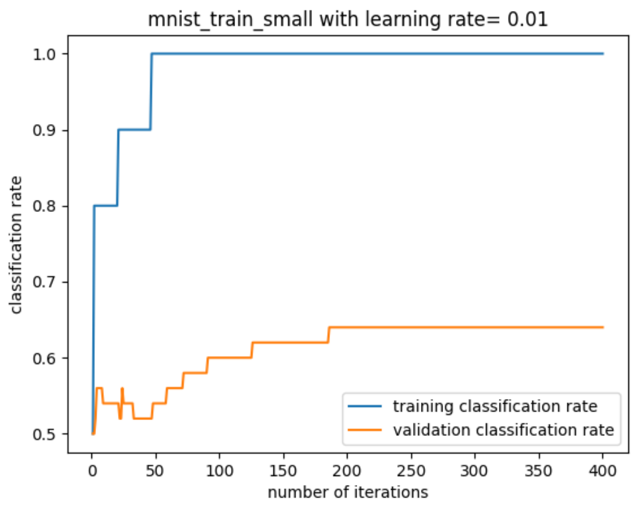
Appendix



```
For the k value in [1, 3, 5, 7, 9]
The corresponding validation accuracy is [0.82, 0.86, 0.86, 0.86, 0.84]
The corresponding test accuracy is [0.88, 0.92, 0.94, 0.94, 0.88]
```

Figure 1


mnist_train with learning rate= 0.1
classification rate
number of iterations
training classification rate
validation classification rate


mnist_train_small with learning rate= 0.01
classification rate
number of iterations
training classification rate
validation classification rate