CAP 5415 Fall 2023

Programming Assignment 3

Submitted by: Ehtesamul Azim

UCFID: 5460629

Objective

1. To implement an autoencoder and a convolutional autoencoder and compare the results along with the number of model parameters
2. To implement k-nearest neighbor algorithms with different numbers of neighbors and compare the accuracies

Autoencoder and Convolutional Autoencoder

Code output

The output of the python file ‘autoencoder.py’ is given below

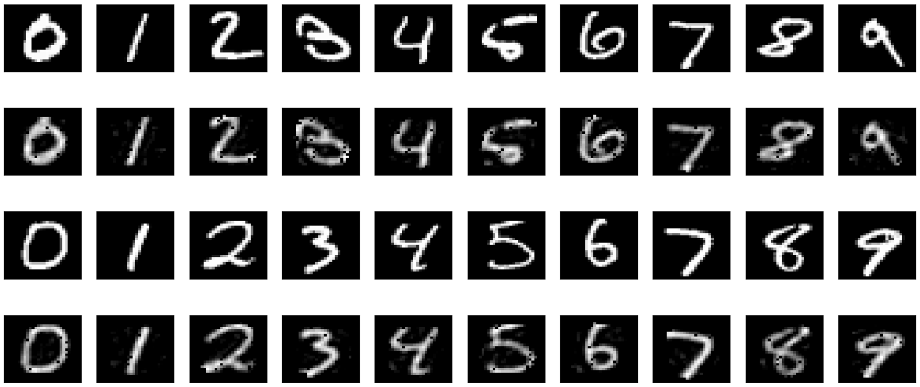
```
running on: cpu

Autoencoder
number of autoencoder params: 468368
epoch: 1/20, train_loss: 0.0469
epoch: 2/20, train_loss: 0.0210
epoch: 3/20, train_loss: 0.0167
epoch: 4/20, train_loss: 0.0149
epoch: 5/20, train_loss: 0.0139
epoch: 6/20, train_loss: 0.0132
epoch: 7/20, train_loss: 0.0125
epoch: 8/20, train_loss: 0.0119
epoch: 9/20, train_loss: 0.0116
epoch: 10/20, train_loss: 0.0113
epoch: 11/20, train_loss: 0.0111
epoch: 12/20, train_loss: 0.0109
epoch: 13/20, train_loss: 0.0107
epoch: 14/20, train_loss: 0.0105
epoch: 15/20, train_loss: 0.0104
epoch: 16/20, train_loss: 0.0103
epoch: 17/20, train_loss: 0.0102
epoch: 18/20, train_loss: 0.0101
epoch: 19/20, train_loss: 0.0100
epoch: 20/20, train_loss: 0.0099
```

```
 CNN Autoencoder
number of params of CNN autoencoder: 2349
epoch: 1/20, train_loss: 0.0891
epoch: 2/20, train_loss: 0.0339
epoch: 3/20, train_loss: 0.0228
epoch: 4/20, train_loss: 0.0175
epoch: 5/20, train_loss: 0.0145
epoch: 6/20, train_loss: 0.0127
epoch: 7/20, train_loss: 0.0115
epoch: 8/20, train_loss: 0.0107
epoch: 9/20, train_loss: 0.0100
epoch: 10/20, train_loss: 0.0096
epoch: 11/20, train_loss: 0.0092
epoch: 12/20, train_loss: 0.0089
epoch: 13/20, train_loss: 0.0086
epoch: 14/20, train_loss: 0.0084
epoch: 15/20, train_loss: 0.0082
epoch: 16/20, train_loss: 0.0081
epoch: 17/20, train_loss: 0.0079
epoch: 18/20, train_loss: 0.0078
epoch: 19/20, train_loss: 0.0077
epoch: 20/20, train_loss: 0.0076
```

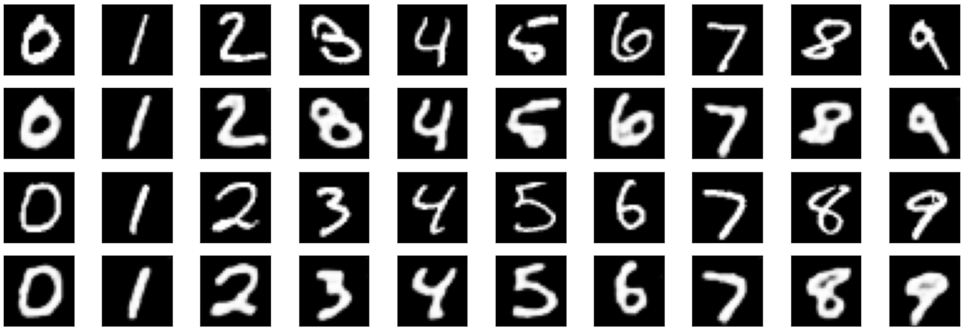
So the training loss goes down in both cases as training goes on. Autoencoder has 468k+ parameters and the CNN one has only 2349 parameters, and still the CNN autoencoder has lower training loss. This shows that in terms of our used evaluation metric MSE, CNN performs better

Representation and Reconstruction

This shows Autoencoder's performance on two sets of original and reconstructed images for each digit.



This shows the CNN autoencoder's performance on two sets of original and reconstructed images for each digit.



In both cases, rows 1 and 3 are the original images and rows 2 and 4 show the reconstructed ones. From these, we can say that the autoencoder reconstructions contain grains similar to salt and pepper noise. On the other hand, although the CNN version has smoother images, it doesn't have such grain.

K-nearest neighbors

(I know the question said to test with n_neighbors with 3 5 and 7 but I checked with other values as well to understand the model behaviour better so I am showing those results here)

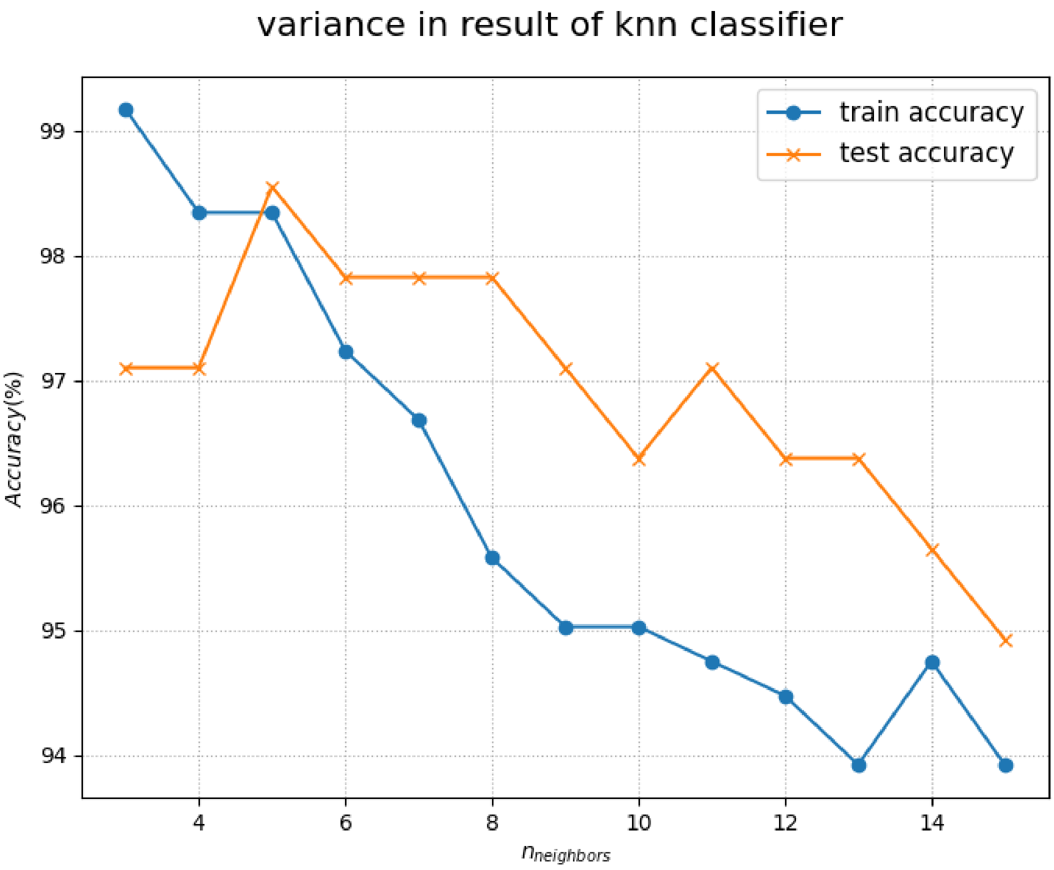
Code Output

```
KNN result for n_neighbours = 2
Train accuracy: 0.9923195084485407
Test accuracy: 0.9818181818181818


KNN result for n_neighbours = 3
train accuracy: 99.23195084485407
Test accuracy: 98.38383838383838

KNN result for n_neighbours = 4
train accuracy: 99.15514592933948
Test accuracy: 98.18181818181819

KNN result for n_neighbours = 5
train accuracy: 98.84792626728111
```

```
Test accuracy: 98.18181818181819

KNN result for n_neighbours = 6
train accuracy: 98.77112135176651
Test accuracy: 97.97979797979798

KNN result for n_neighbours = 7
train accuracy: 98.84792626728111
Test accuracy: 98.18181818181819

KNN result for n_neighbours = 8
train accuracy: 98.61751152073732
Test accuracy: 98.18181818181819

KNN result for n_neighbours = 9
train accuracy: 98.54070660522274
Test accuracy: 97.97979797979798

KNN result for n_neighbours = 10
train accuracy: 98.46390168970814
Test accuracy: 97.77777777777777

KNN result for n_neighbours = 11
train accuracy: 98.38709677419355
Test accuracy: 98.18181818181819

KNN result for n_neighbours = 12
train accuracy: 98.31029185867895
Test accuracy: 97.77777777777777

KNN result for n_neighbours = 13
train accuracy: 98.15668202764977
Test accuracy: 98.18181818181819

KNN result for n_neighbours = 14
train accuracy: 98.15668202764977
Test accuracy: 97.77777777777777

KNN result for n_neighbours = 15
train accuracy: 97.6958525345622
Test accuracy: 98.38383838383838
```

So the best training accuracy is achieved with 2 neighbors only and the best is achieved with 5 neighbors. Going beyond that, the performance always gets worse.

Accuracy vs number of neighbors plot



So the number of neighbors affects both the training and the testing performance.