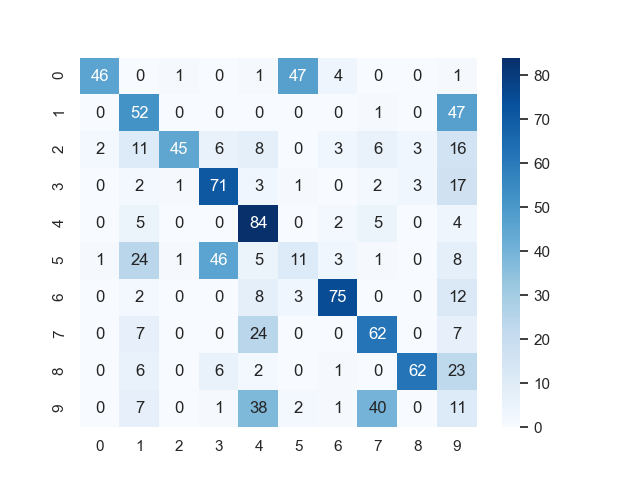
1. k-means

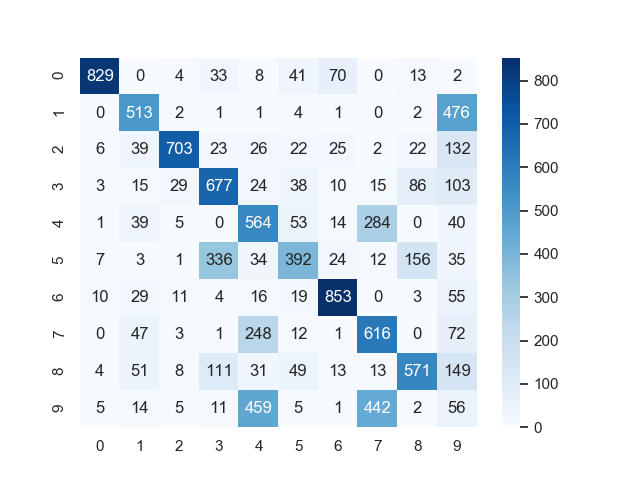


Accuracy: 0.519

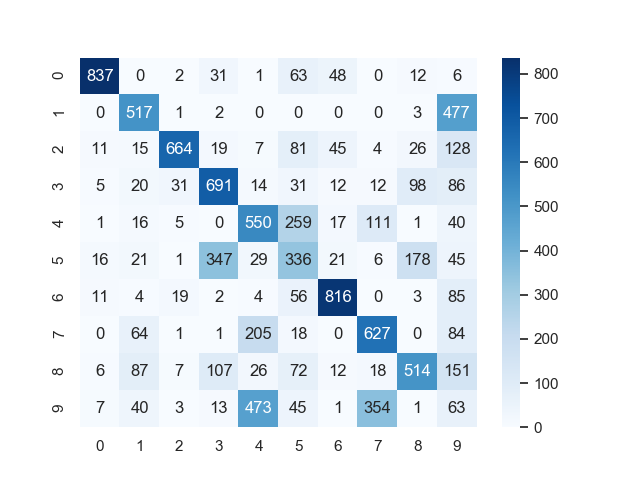
The accuracy compared to the first assignment is rather low. I’m not surprised though given this is done through unsupervised learning as well as clustering in

2.

a. K-means with feature vectors from autoencoder (trained 20 epochs). Using all 64 feature vectors achieves better accuracy than k-means run on the unaltered MNIST vectors. The errors in classification appear rather consistent though, even if there are less of them. For example: 1’s are misclassified as 9’s about as frequently (as in percentage of misclassifications) in this example as they are in the above example – this goes for many of the other misclassifications as well.

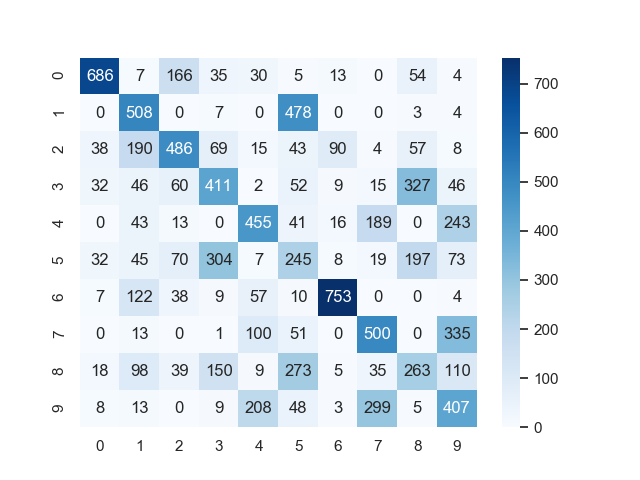
Accuracy = 0.5774

b. K-means with PCA using 10 PC’s generated from the same autoencoder’s output as above. K-means with 10 PC’s achieves slightly worse accuracy than using all the feature vectors from the autoencoder. I wouldn’t say this with certainty however, as it’s noted that k-means is highly dependent upon the initial random seed – which is why the k-means algorithm is usually run multiple times and the best fit picked out of those runs. Because that wasn’t done here it’s hard to say with certainty that 10 PC’s from the autoencoder will usually be slightly worse.



Accuracy = 0.5615

k-means with PCA using 4 PC’s



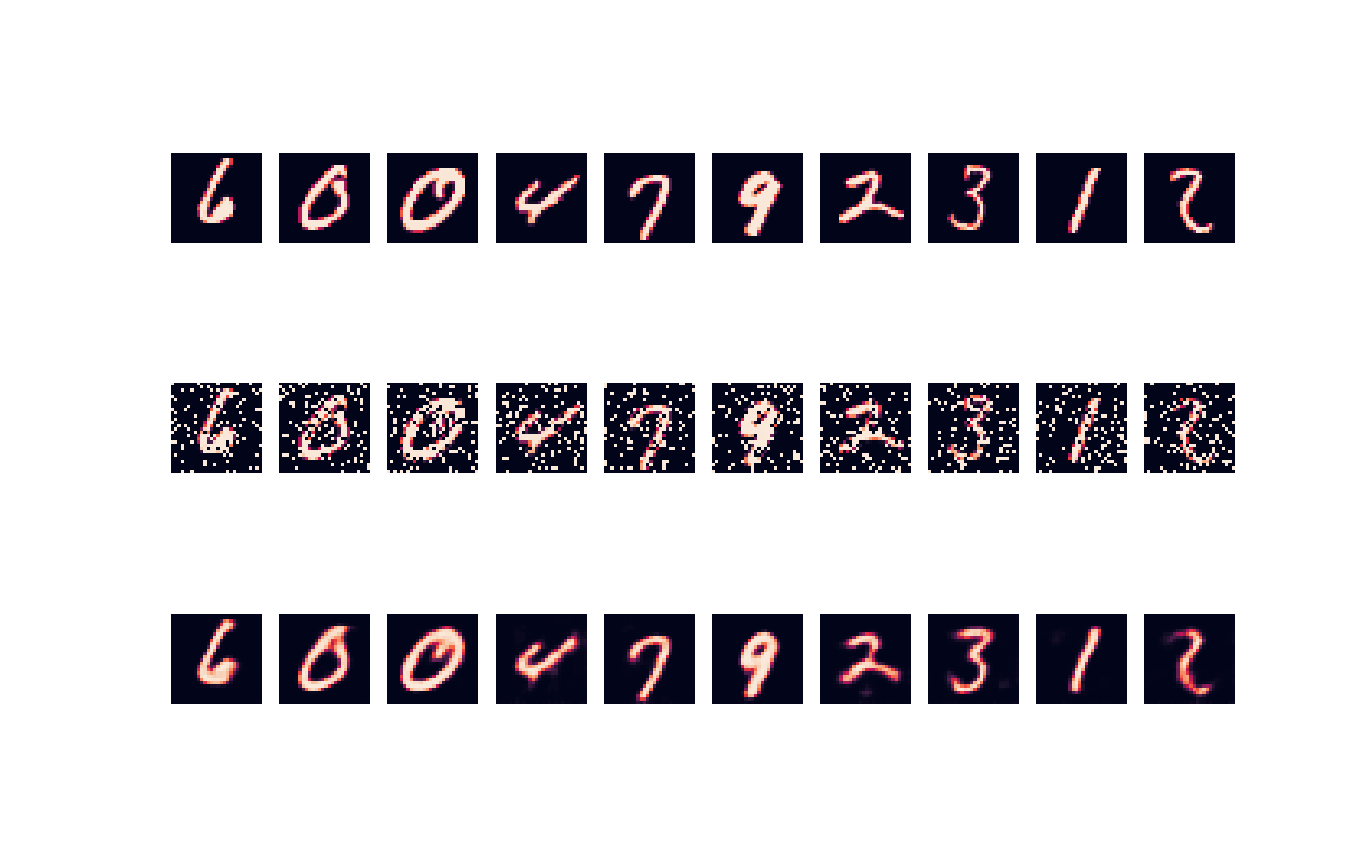
Accuracy: 0.4714

3.

a. Using salt and pepper noise method linked in doc.

b. ratio of noise: ~1/3rd of the pixels are changed to noise. (there’s not check for redditive pixels being chosen so the maximum number of pixels changed is 1/3rd – regardless though, it’s roughly 1/3rd of pixels that are changed)

c.

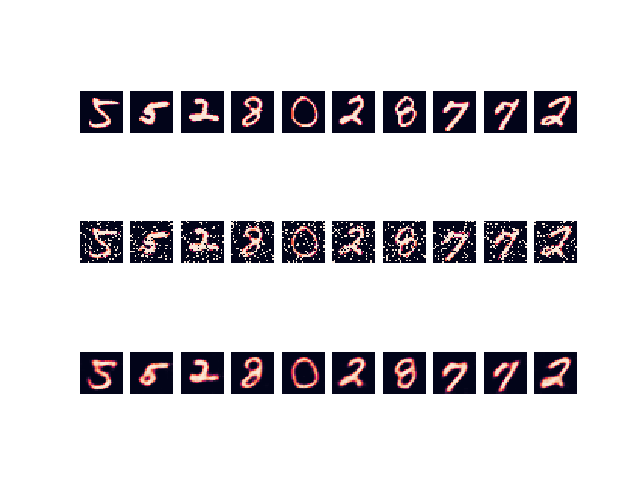


d.

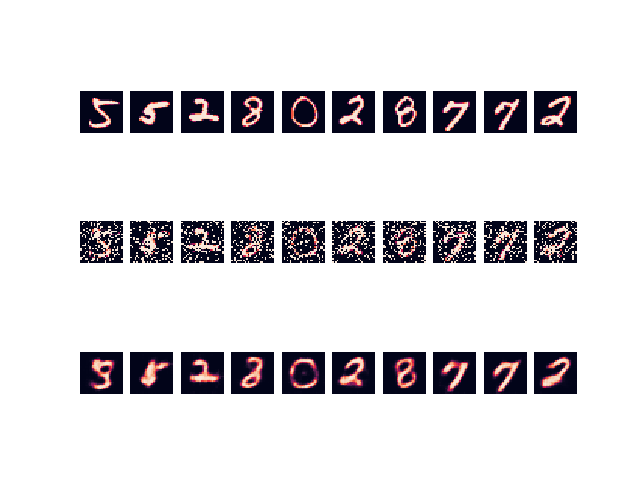
Noise ratio =



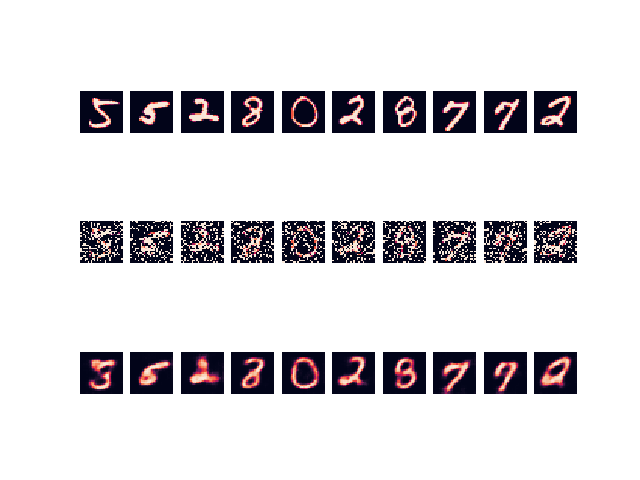
Noise ratio =



Noise ratio =



Noise ratio =



Noise ratio =

