November 19, 2022
Homework 5
Jaren Glenn - A02298742#

Problem 2

1. Mini-batch size *largely* only affects training time. It's a trade-off, though - smaller mini-batch sizes are much easier to compute, but if they're *too* small they start to have a negative impact on the accuracy because the computed gradient might be wildly inaccurate. As the complexity increases, it becomes more and more necessary to find an m that helps cut down computation while not affecting accuracy too much. For instance, Resnet-151's ideal mini-batch size is likely lower than Alexnet's simply due to the fact that Alexnet is a much smaller network and can afford to search for that little bit of improved accuracy that comes with a larger m.

2. The first thing to try changing would simply be the learning rate - it could be as simple as having a value that's too high. If η is is too high, then the gradient will just keep jumping around the function and the network won't learn anything.

   If the network takes a long time to train, another thing to do would be to reduce the problem down to its simplest components. If you can just focus on distinguishing between two or three classifiers on less items, you can experiment with parameters and architectures much faster. Another potential problem might be that the network simply doesn't have enough parameters to learn the function properly, so adding more nodes / another layer could improve things.

   Regularization won't really help if the network is just randomly guessing, so that should wait until the network is somewhat working. Once there, regularization can help improve accuracy further.

Problem 3

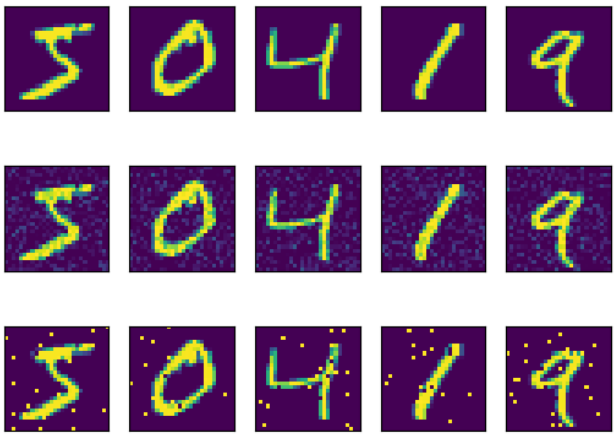
1. The principal components of the data are the eigenvalues of the data's covariance matrix, simply performing SVG on the data matrix as a way of extracting the principal components is entirely valid.

2. PCA is a method of dimensionality reduction can be viewed as minimizing the linear reconstruction error $\|X - \hat{X}\|_2^2$ where $\hat{X}$ is the reconstruction and X is the original data. This is, in essence, what an autoencoder is designed to do as well. For instance, a linear decoder with MSE loss will minimize $\arg\min_w \mathbb{E}[\|x - w^T wx\|^2]$ - exactly the same as PCA. Other variations on autoencoders may not give the exact formula as PCA, but they offer similar functionality to it (and sometimes more, such as with nonlinear en/decoder functions).

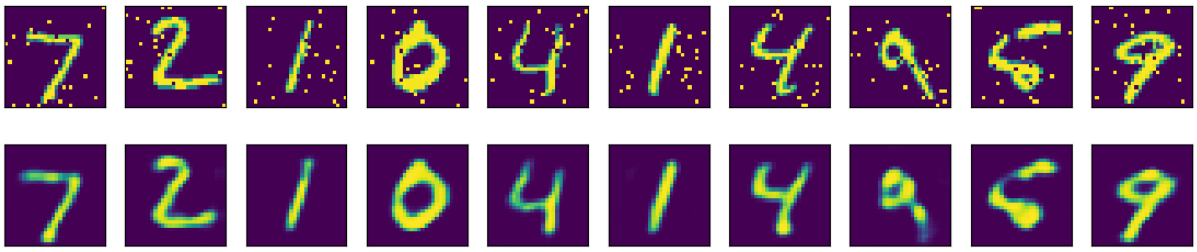
3. As the names suggest, linear autoencoders are built using connected linear layers while convolutional autoencoders are built using convolutional layers. Essentially, the network learns to minimize the reconstruction error by learning the optimal filters for the convolutional layers. They focus more on the general features of the image rather than learning it on a pixel-by-pixel basis. Linear autoencoders are a lot closer to the more general PCA.

Problem 4

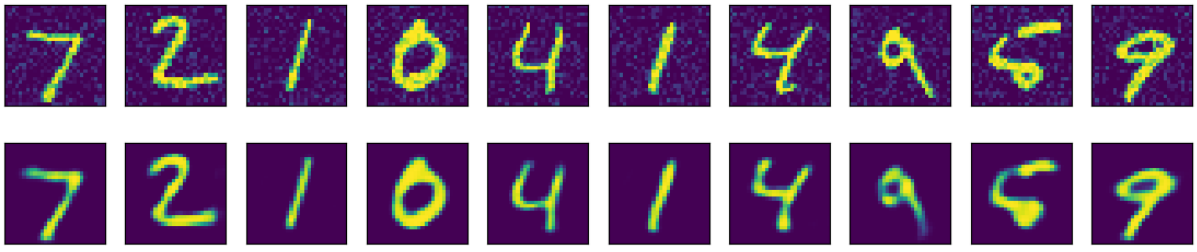
Some samples of applying noise to the dataset:



My network achieved a training error of 0.012898 and validation error of 0.012553 with salt and pepper noise, and a training error of 0.009170 and a validation error of 0.009416 with Gaussian noise.



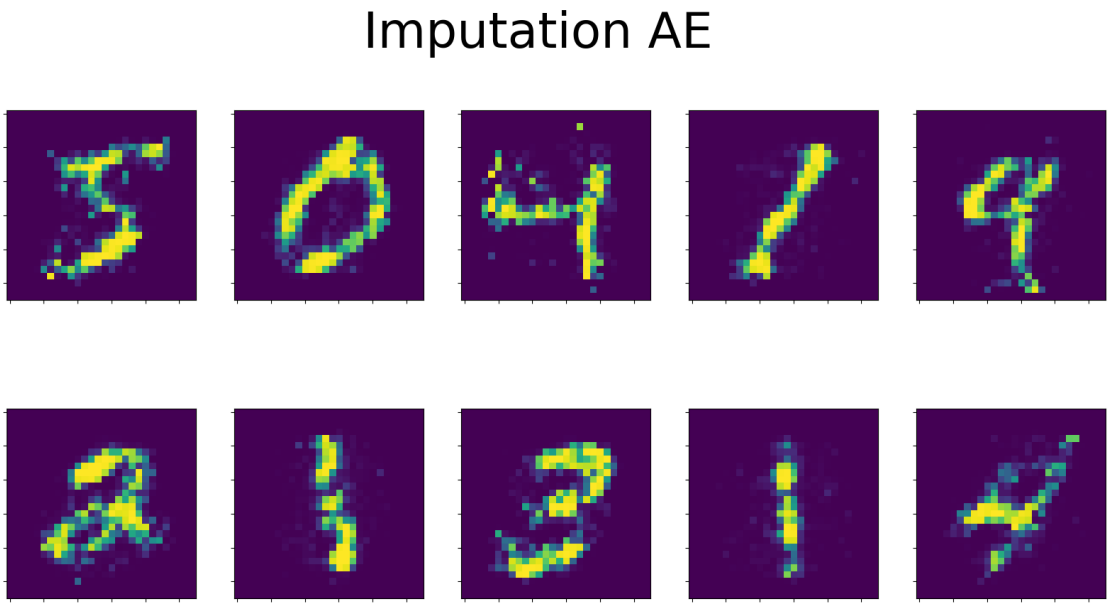
Running the above images on the salt and pepper network gave a test error of 0.010325; while it does have some issues on a couple images for the most part the de-noising is successful.



Running the above images on the Gaussian network gave a test error of 0.008267. This is closer to the original images compared to the salt and pepper network, but still struggles on the same parts of the same images.

Problem 5

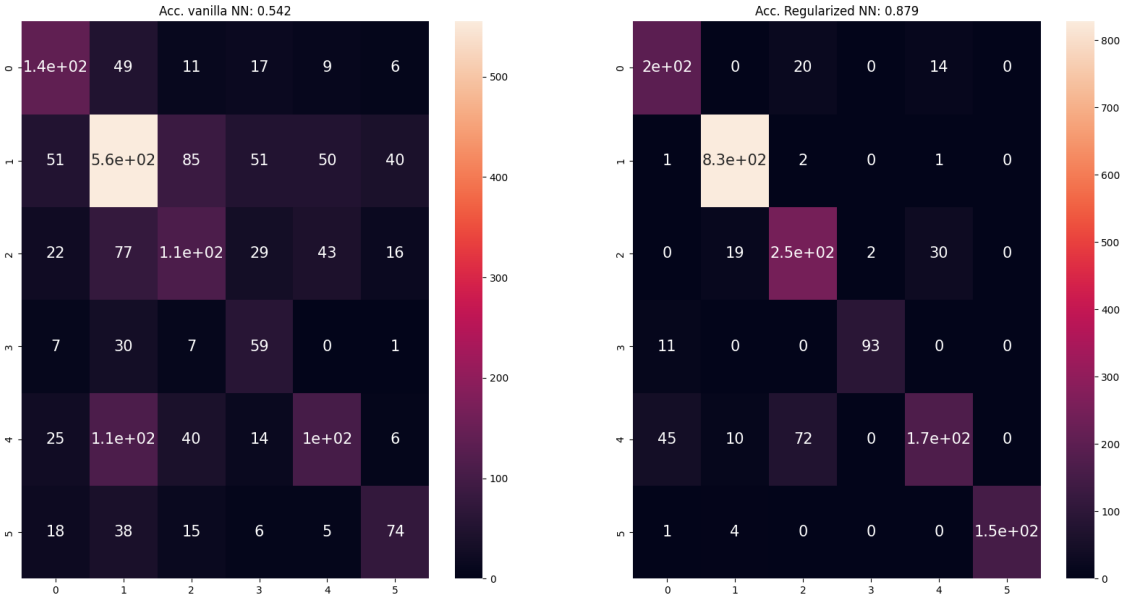
After training my imputation AE these are some of the results:



Imputation Method	MSELoss
Autoencoder	0.0336
K-th Nearest Neighbor	0.0179
Simple Mean	0.0399
Simple Most Frequent	0.0662

While the AE didn't perform as well as the KNN imputation, it performed slightly better or on par with the simple mean and much better than the simple most frequent. It's interesting though that the neural network performed pretty signficantly less well than the KNN imputation; the AE has a lot more missing areas and noise around the edges.

Problem 6



The vanilla NN (left) ended up with a test accuracy of 0.5043% and the regularized NN (right) had a test accuracy of 0.8523%. This is clear in the visualizations too - the right image is a lot more confident and the values are more concentrated in the correct areas.