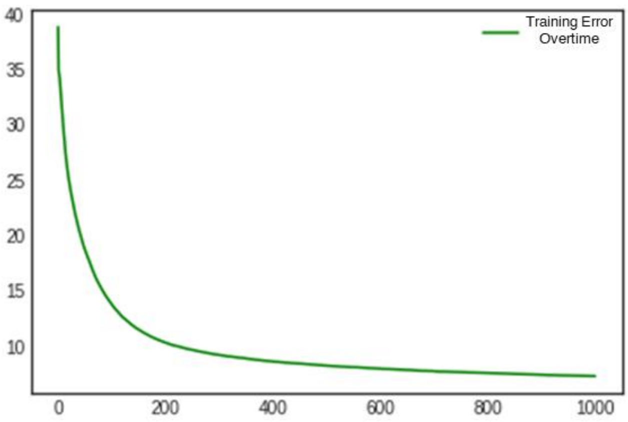
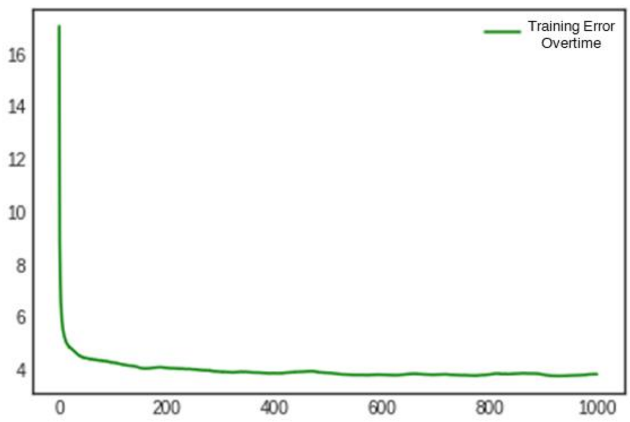
AI 502: Implementation → K-Sparse Autoencoders
Sangyoon Lee (20174375)

Concept Experiment

Authors argued that encouraging sparsity is very effective in classification tasks. Further, compared to simple conventional sparse coding, the performance and efficiency is much stronger. Therefore, in this section, I compared the training losses from simple conventional sparse coding and K-Sparse autoencoder. Dataset used for this and next experiments are **MNIST dataset** and every training had **1000 epochs**.



Simple Sparse Coding Error



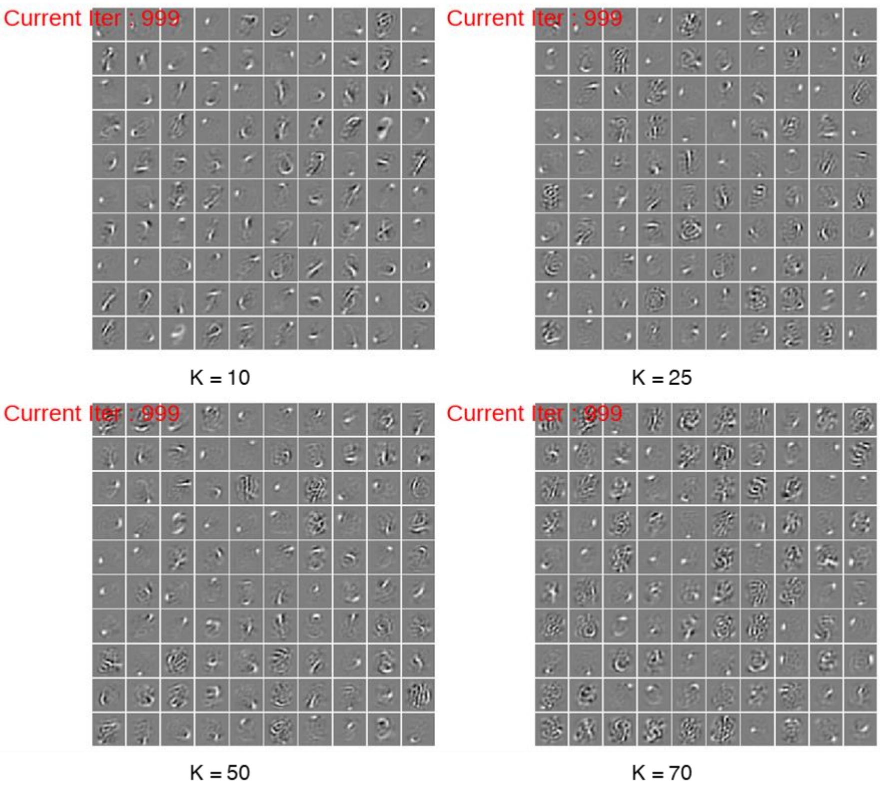
K-Sparse with K=25 Error

If you see the loss graph above, **K-Sparse autoencoder had a final error of 3.825 and Simple Sparse Coding had a final error of 7.251**. Therefore, as the authors mentioned in Table 1 of the paper, K-Sparse autoencoder is much better than simple sparse coding. Then, on the next experiment, I tried to demonstrate the effect of K size on learned weights of K-Sparse autoencoder.

Performance Experiment

Here, I tried to replicate Figure 1 of the original paper. According to the authors, K determines the sparsity of the learned weights and such tuning provides practicality of current autoencoder. They found out that as the value of K increases, the algorithm learns more local features. If you see the figure below, although it is not very distinct, **when K = 10 and K = 70 were compared, K = 10 definitely learned more about general features while K = 70 learned specific points of the images.** If you look closely, in K = 10 images, you can somewhat distinguish the target hand writing. If you check the attached video on my experiment folder, the evolution of iterations demonstrates differences in sparsity much more clearly than final images on this figure. Further**, errors for each of the different sparsity was around 4, which were fairly high.** Like Shown in Table 1 of the paper, K = 25 of my experiment did showed the best performance while K = 10 suffered slightly (please refer to attached graphs).

Learned Weights with Contrast Normalized



Discussion

The main limitation of the current experiments is that specific hyper-parameters are not exactly the same as the paper descriptions. Due to limited time availability, I tested with working hyper-parameters for my code only. Further, because it was a little heavy to run 1000 hidden units, I used 100; this might has been the main reason the sparsity was not clearly distinguished.