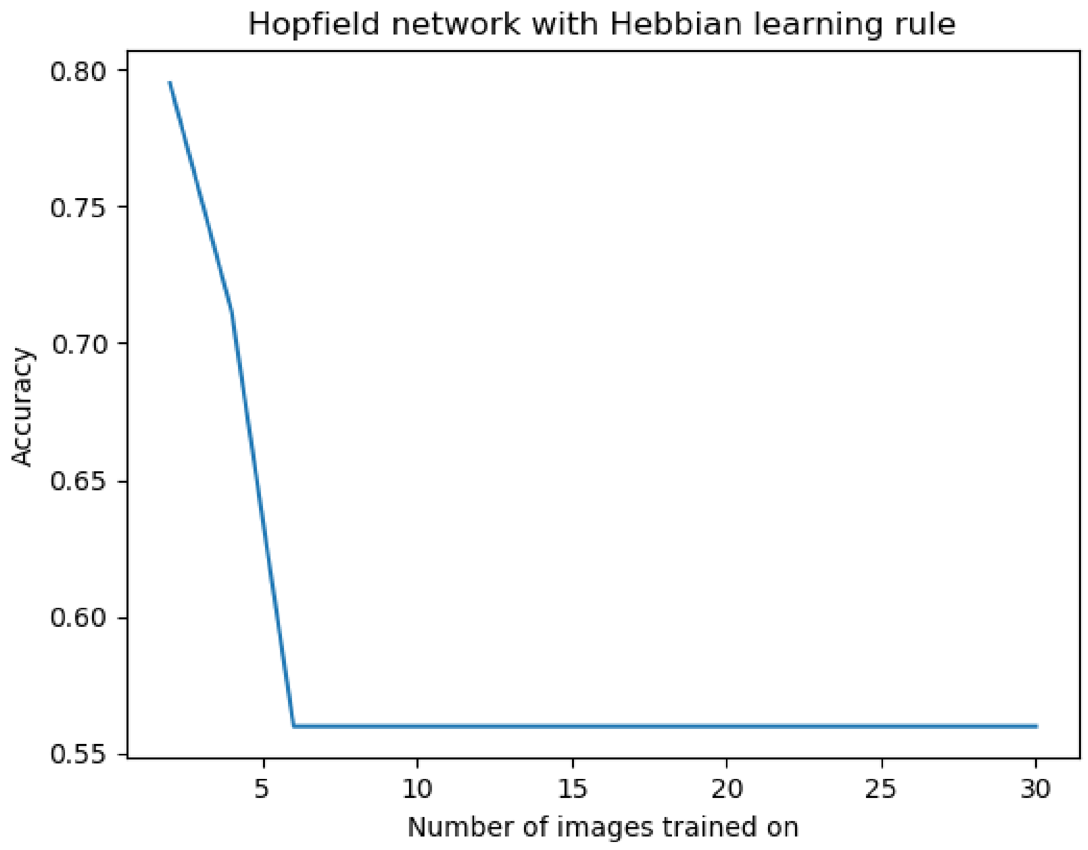
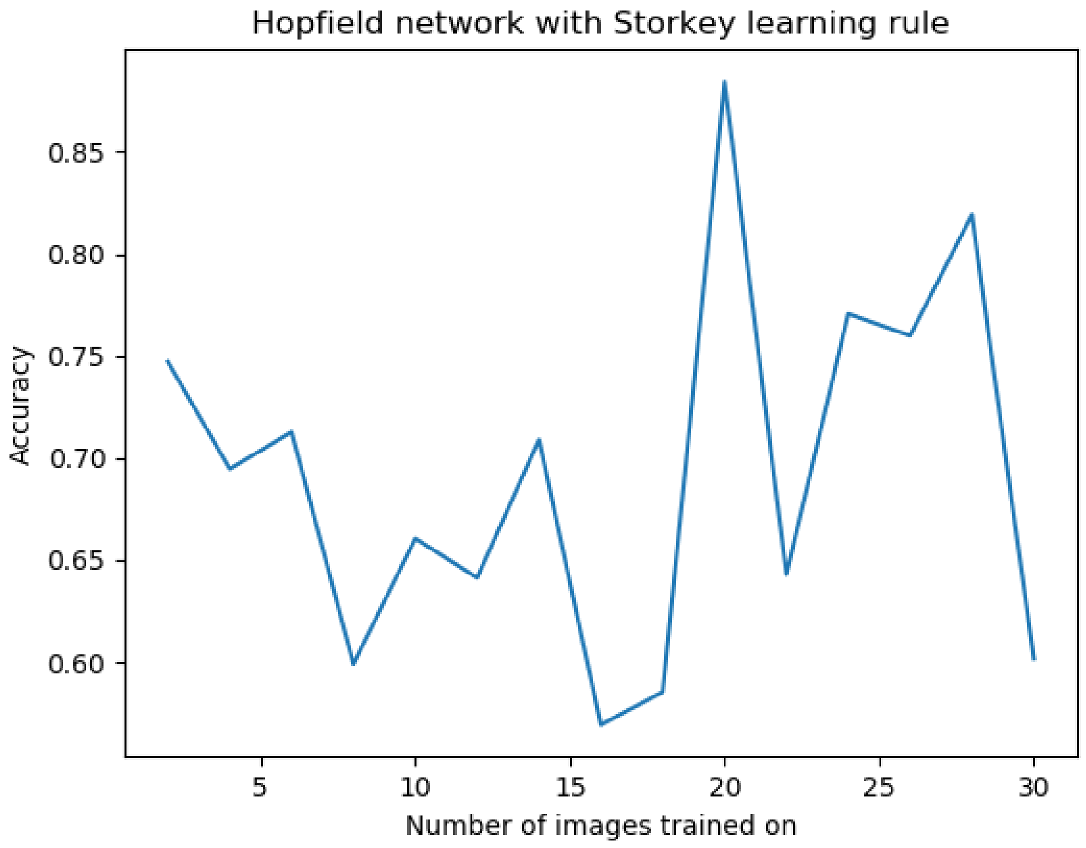
Question 1

1a)



Evidently, accuracy peaks early on, at 2 images. If the network is given too many patterns to learn, it eventually breaks down and fails to model anything accurately, resulting in a model that only ever predicts the prior (the test set is 56% ones and 44% fives, which is why accuracy plateaus at 0.56.) Note that # of training images is always a multiple of 2, as training was done by giving the network 1 five and 1 one, 2 fives and 2 ones, etc.

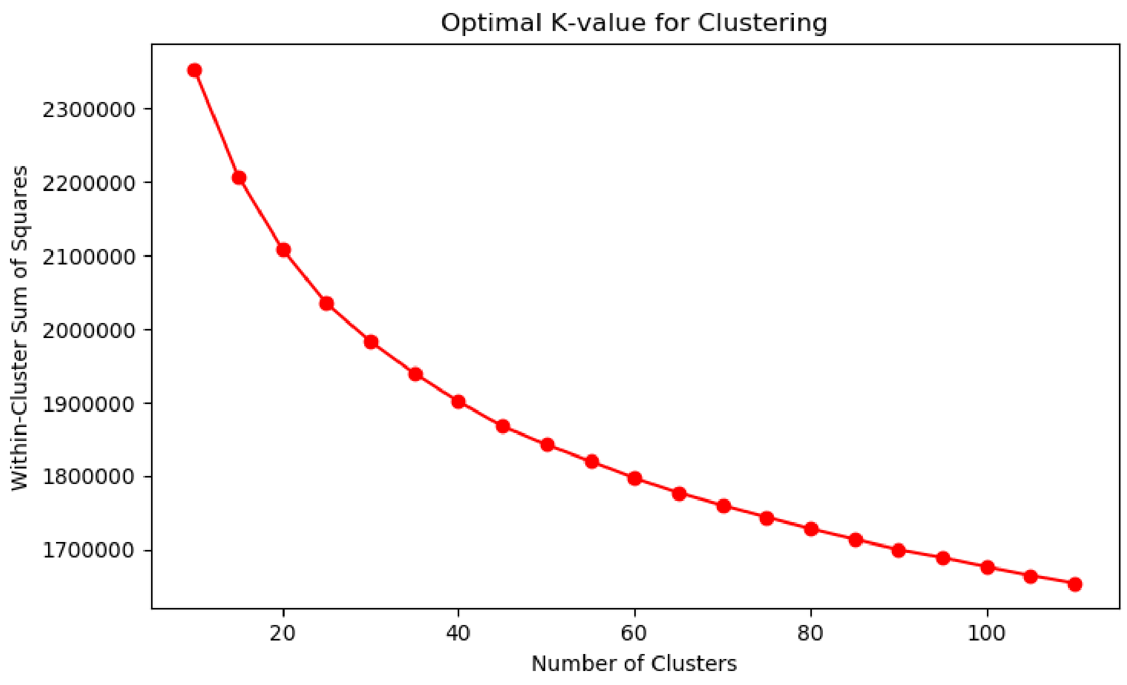
1b)



Storkey's rule appears to be more volatile; accuracy varies to a far greater degree across different numbers of images. However, it is much less susceptible to overfitting, and has a higher peak than the standard Hebbian Hopfield network. Additionally, its lowest accuracy is no lower than the lowest points of the Hebbian variant, and it rarely reaches that point. Clearly, it is capable of storing a larger number of patterns.

Question 2

2a)



The above is a graph of various attempts at k-means clustering on the mnist dataset. Each k-value was iterated over 10 times (the k-means default), with the lowest within-cluster sum of squares (or inertia, per scikit-learn's terminology) being preserved. Using the elbow method, it appears that the rate of descent begins to slow at 50 clusters, meaning that my RBF model will have 50 neurons in its hidden layer.

(N.b.: graph was computed using the optimal_K function in q2.py, which takes a very long time to finish executing. Instructions on how to replicate an approximation to the graph at a much faster pace are in comments inside the file.)

2b-c) Unfortunately, flaws in implementation make the RBF network incapable of performing better than ~11% accuracy.