Lab 3A Report: Obstacles and Solutions:

While implementing the sparse autoencoder, I ran into a few main obstacles. The first was getting the gradient check to pass: my analytical gradients from backpropagation did not match the numerical gradients at first, which usually pointed to a mistake in either the cost computation or the backprop formulas. A second issue was the sparsity penalty: the KL divergence term and its derivative are easy to get wrong (e.g., mixing up the order of arguments or forgetting to add the sparsity term into the backprop delta for the hidden layer). I also had to be careful with shapes when unpacking and repacking the parameter vector into W1, W2, b1, b2, and when computing the average activation rho_hat over the right axis so that dimensions matched in the cost and gradients.

To fix these, I worked through the assignment step-by-step as suggested. I implemented the cost and gradients in stages: first only the squared-error term with weight decay and beta set to zero, ran gradient checking until it passed, then added weight decay and re-ran the check, and finally added the sparsity penalty and its derivative (including the sparsity_delta term in the backprop for the hidden layer). For the KL term, I used the formula from the notes and avoided log(0) by clipping rho_hat away from 0 and 1. When the gradient check still failed, I used a smaller numerical epsilon and double-checked the sigmoid gradient and the order of matrix multiplications. Once the gradient check passed, I trained with L-BFGS and verified that the learned features looked like edge detectors on the image patches.