Using my ulitimate brain I found out the best configuration for training the small neural network to solve the xor problem. The problem with xor is that the space which the perceptron is trying to solve is not lineralibly sepreable. I found that the best neural net work that solved the problem was one that had two inputs and one out put. As the xor opperation has. In the middle i had a hidden layer that had four inputs/outputs. Playing aronudu with the learning rate solved gave some intresting results. With a learning rate of 0.01 an average of 50,000 rounds was required to have the loss decreese to .05 which i used as the max amount of loss accepted. The neural network was able to correctly solve the problem up to around .20. To make sure that the neural network would properly solve the problem each time it was run I made sure the threshold was low enough that the outpuut wouldu be good. Validate the model I added some test cases to the program. These test casses validated the model that wolud produced my the neural network. In future work the sample size of the trainng set would be increased. Tihs would probelbly address thecasses that i encounterd where the neural network would not be able to decrease the loss around 0.55 to .39 it would suddenly be learning with a stedaly decreasing amount less and less each time this. Usually when running the neural netword the it would have slightlily less good learnign rate when the loss was around 0.45. This problem was the worse when the n\_layer1 was not set to 4. Less or more tended to break the model about half the time. With n\_layer1 set to 2 it would sucsees about half the time. But setting it too high would not work either. When set to 10 the neural network would not work at all. 4 turned out to be the best value for the value of n\_layer1. I also tryed to us the random uniform distribution of the generation of the weights of W2 but that also made the model perform suboptomitly. I found that initilizing the weights to zero for the second layer produced the best results for clasiffing the xor problem. While this was true for the second layer it is not true for the first layer. in the first layer not initalizing to a random uniform distribution resulted in the model breaking. This may suggest that when the model performs sub optomly that the inital weights for the first layer are not a good fit for the model. This is an intresting result that I did not expect when I started this project. Inctrestingly initiliing the biasize on the first layer to either zero or one did not matter for this neural network. When either where used they output aronud the same average result.