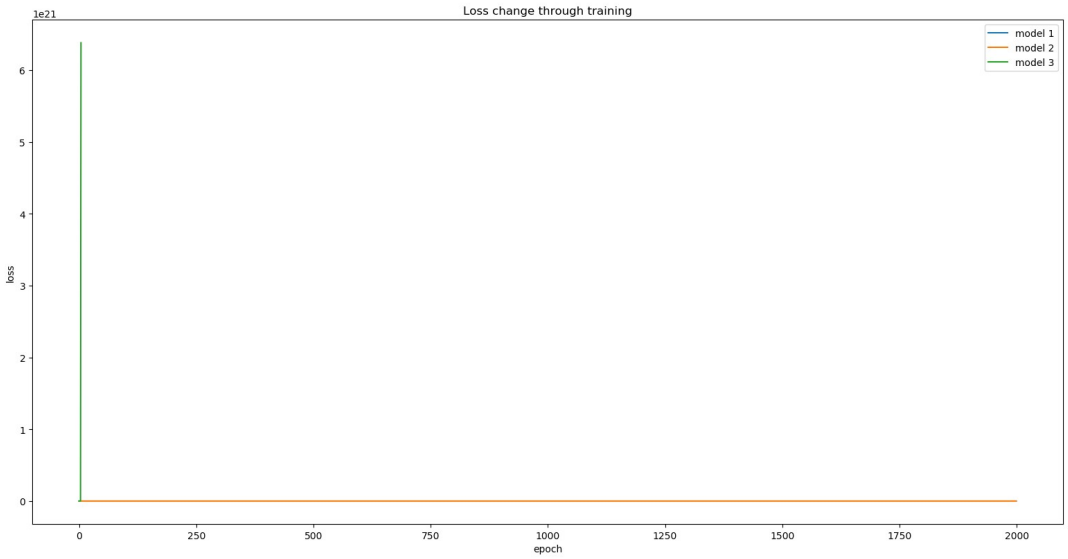
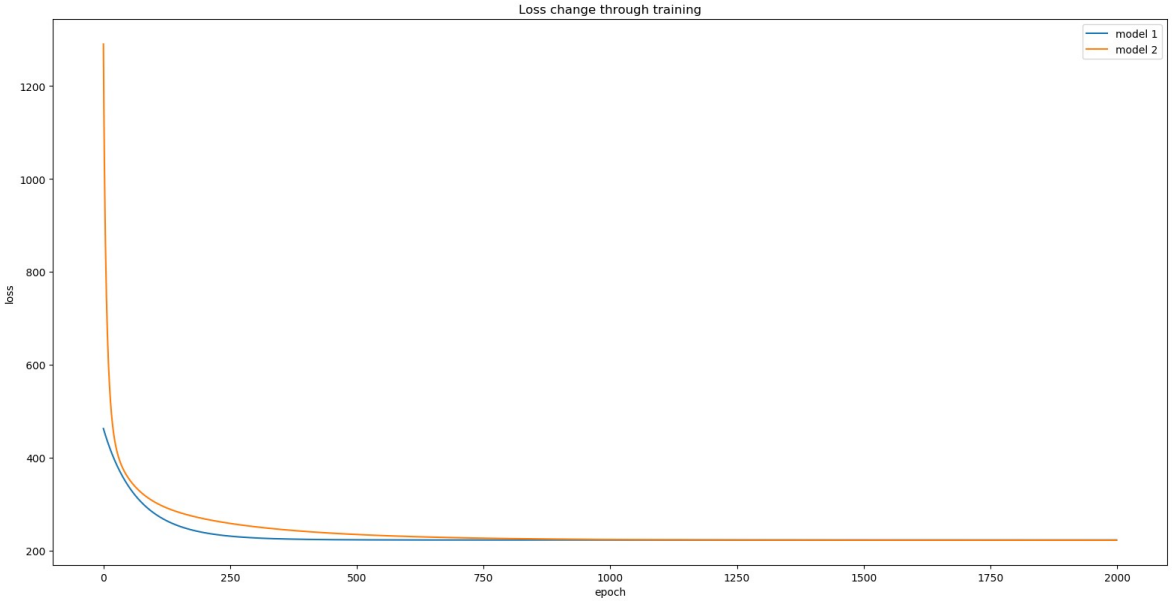
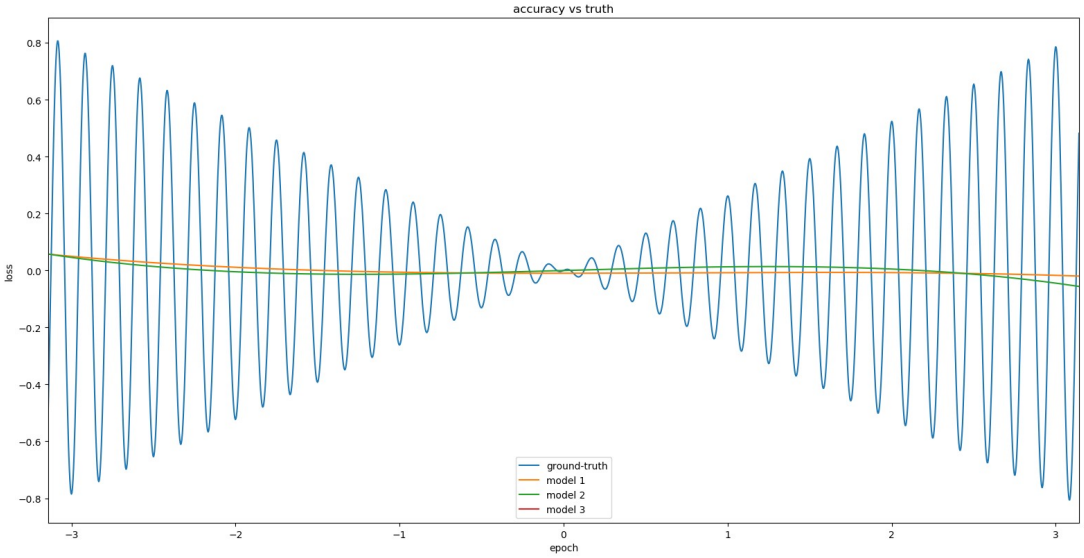
1-1. The function I chose was cos(12*math.pi*x)/12*math.pi*x . I chose this as it was a simple variation of the example function given in the homework slides. The models I used are also direct adaptions of the 3 example models from slide 6 of the homework slides. The graphs of the accuracy and loss are below of my models are below.



The value of the loss for model 3 is so high that it makes it impossible to see the others, so when we cut it from the graph we see:

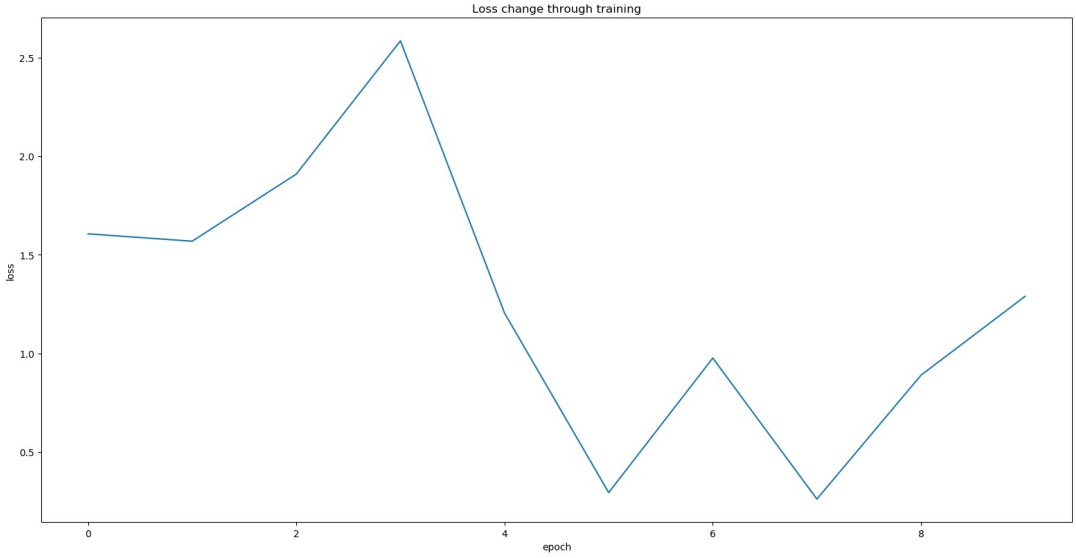


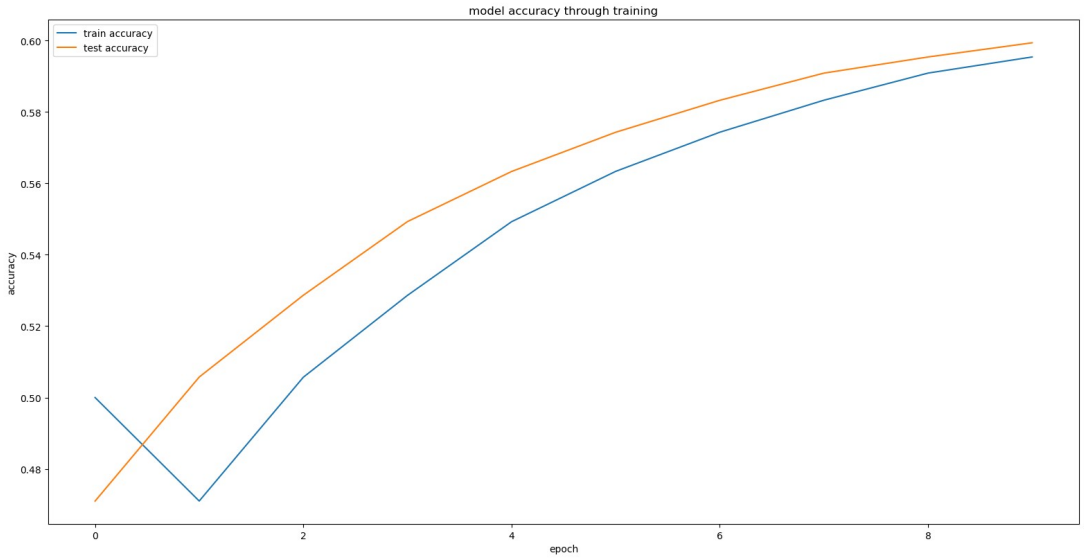
and now for accuracy:



As we can see, these models are highly inaccurate on predicting my function. There is very high loss (ludicrously high in the case of model 3) and low accuracy. Model 1 minimizes loss the most and seems to perform the best. More data/training time may have been necessary than what I gave them; I only trained them for about 2000 epochs. These models may have been sufficient to learn the sin and sign functions given in the slides, they clearly fail to work very well on the scaled cos function I used.

1-2. For the task I chose CIFAR-10. My model is pretty simple; 1 convolution layer, followed by a pooling layer, followed by another convolution layer, followed by 3 fully connected "linear" layers. The learning rate is 0.001, and the model is optimized with stochastic gradient descent. Due to the computing resources available to my computer, I chose to only train it for 10 epochs, to keep running time down to 5-10 minutes. Below are the training loss and training accuracy graphs.

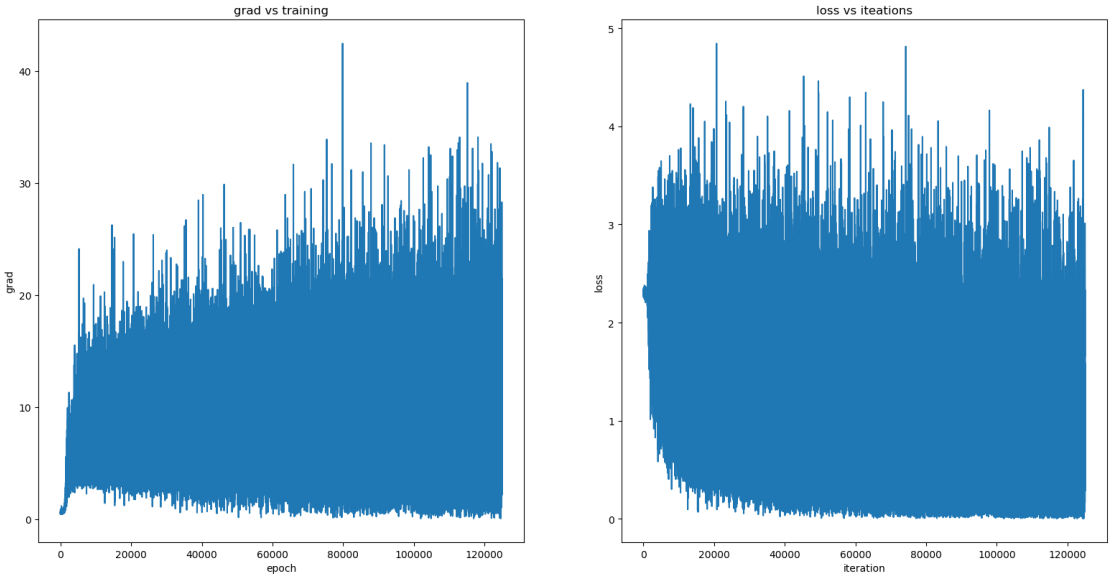




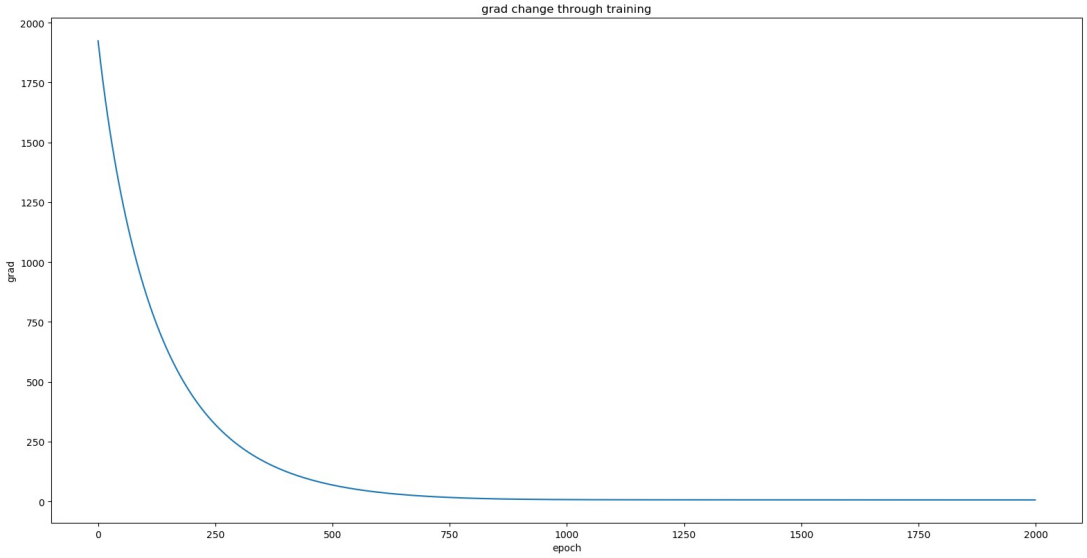
I achieve just below 60% accuracy in both train and test, which isn't too bad given the simple nature of my model and small data set.

2-1. While I attempt PCA in my code, I'm unable to transform the data to create graphs from it.

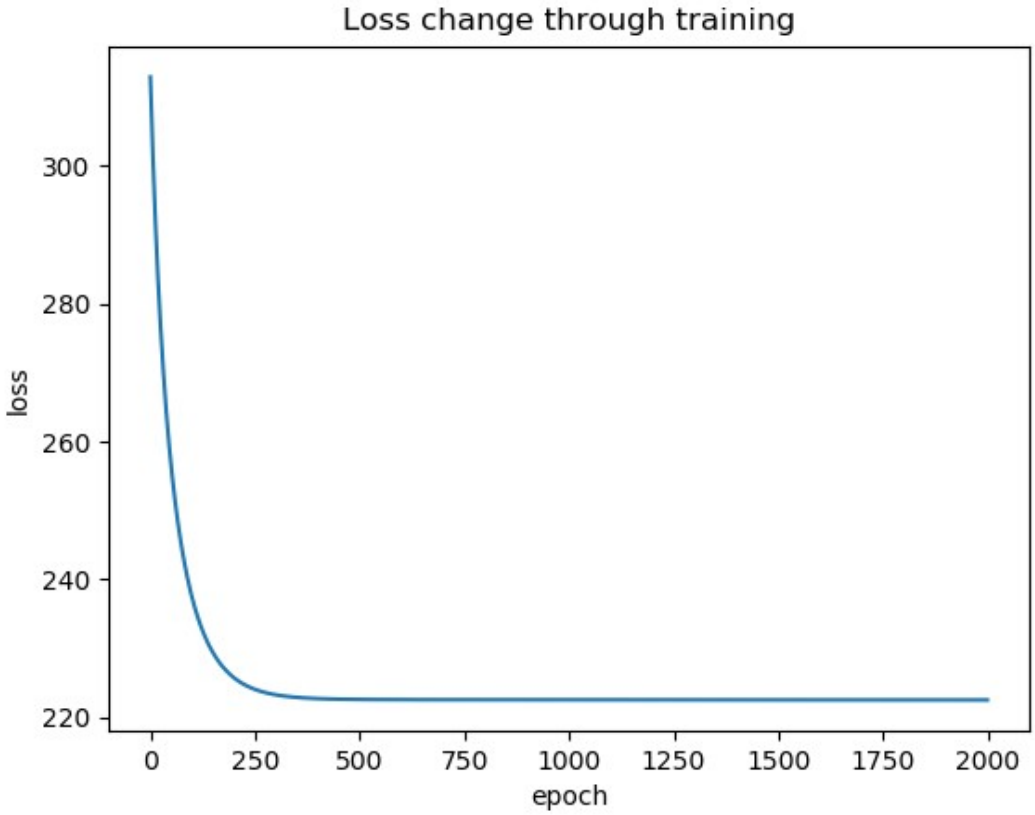
2-2. Below are the graphs showing the grad vs iterations and loss vs iterations for my CIFAR model: As we can see, the values of both are highly volatile. The model is not trained long enough for the optimizer to really find a local maxima through gradient descent. Both graphs remain volatile and show no clear trend.



We can now look at the grad graph for the model 1 we used earlier on the cos function:

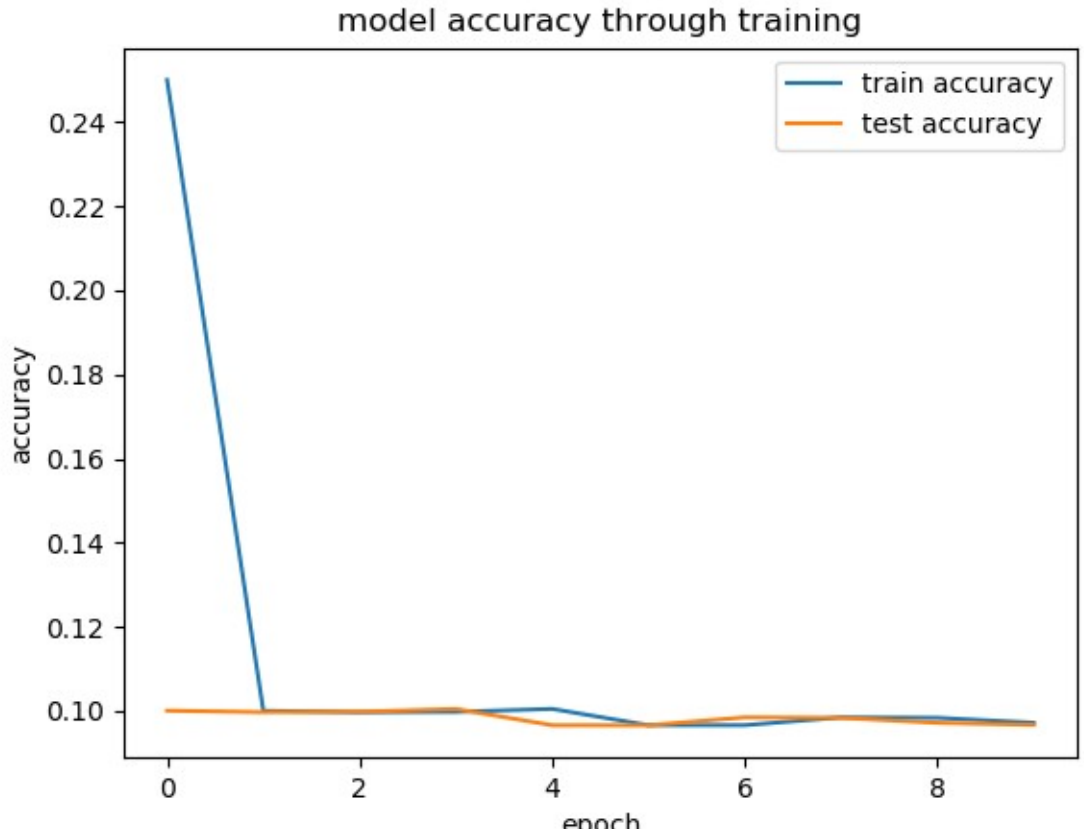


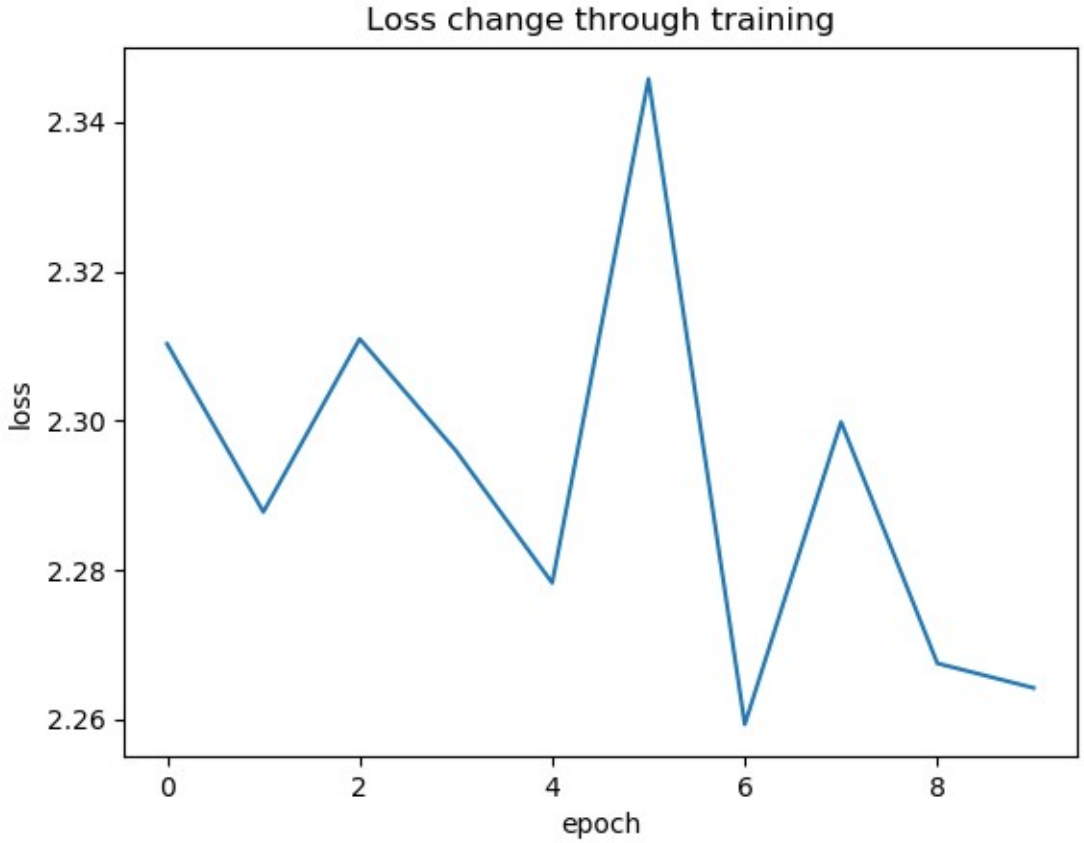
and the loss graph for model 1:



We can see a much clearer trend of optimization over training, although the value of the loss remains quite high, and the gradient does level off fairly early on around epoch 300. With some tweaks its likely this would optimize nicely.

3-1. For the random labels test, I use the same model on CIFAR-10 with random labels. I use a learning rate of 0.001, and optimize with stochastic gradient descent. Graphs of the loss and accuracy after 10 epochs are below:





As we can see, having completely random labels prevented the network from learning anything at all over our 10 epoch training period, as the relationship between label and picture was lost. More time/epochs might have allowed it to learn something about our training data, but that would have of course led to to incorrect testing results, as the labels between both sets no longer matched. Loss was highly volatile throughout the whole process, although a slight downward trend is visible.