Neural Network for MNIST Dataset Report

-Abhinava Sikdar(2017MT20724)

Some Facts About My Model

Which activation function have I used & why?

The three main activation functions that I have considered are sigmoidal, tan hyperbolic and ReLU(Rectified Linear Unit).

While using sigmoid has benefits like the activation doesn't blow up, however there is a tendency of the gradient to vanish as we go deeper into the network. Such a trade off exists for ReLU too. While the activation of ReLU can blow up, it doesn't share the problem of vanishing gradients unlike the sigmoid function. Also while ReLU is more efficient in terms of computation(one comparison only), it also faces what is know as "Dying ReLU" in which if a lot of activations become zero, most of the neurons in further layer give an output of zero and thereby prevent learning(can by overcome by using Leaky ReLU).

The properties of the tanh and the sigmoid function are pretty similar, however tanh seems to give better results when I used the same as my primary activation.

It has been seen empirically that ReLU converges faster, is easier to compute and is optimal for deep networks (since no vanishing gradients), hence I have chosen ReLU as my primary activation function.

Why have I used softmax in last layer?

Since I have used ReLU as my primary activation function which is unbounded, I have used softmax function to convert it to a discrete probability distribution over the values of 0,1,2…9 instead of raw unbounded values for these digits.

Why have I used cross entropy cost function?

While MSE is preferred for regressions, for classification tasks Cross-Entropy Error is preferred. This is because in classification, the output is either right or wrong and when it is the later, we want to penalize the classifier heavily, however in regression we don't want to penalize heavily if the prediction is in some range of the data label.

How have I initialized the weight matrices and bias vectors?

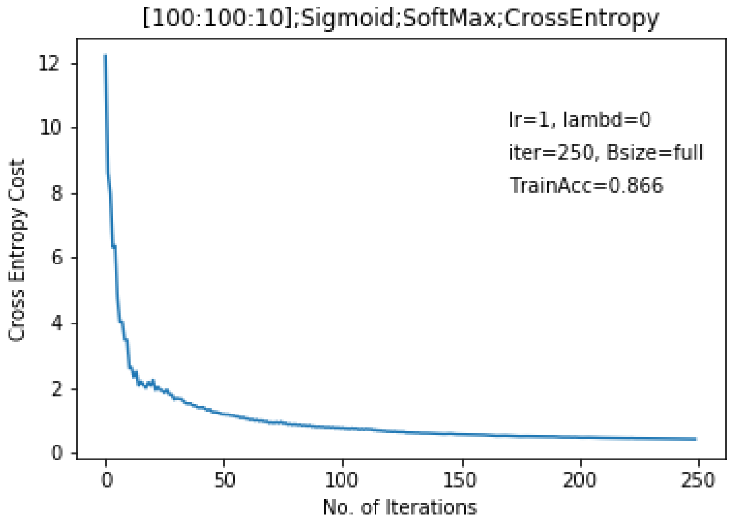
I have initialized all the bias vectors to zero and am sampling from Standard Normal Distribution(or N(0,sigma^2) sometimes) to initialize the weight matrices for the network

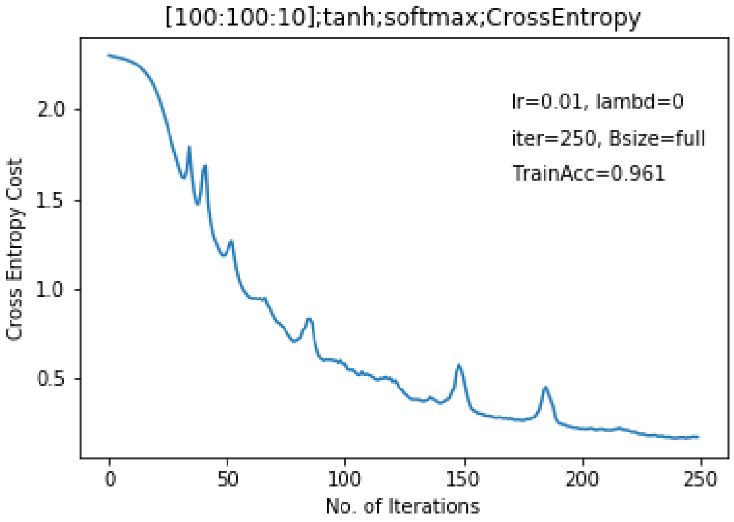
What are the parameters that I have used for the graphs below?

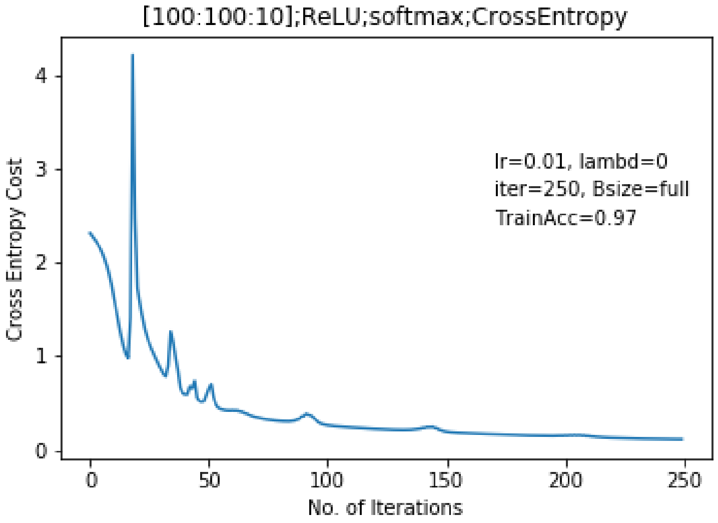
Unless otherwise mentioned and except for the parameter/function that I am varying, I have generally used the model with 2 hidden layers and an architecture as [784:100:100:10] and the full training data of 6500 examples as the batch size. I have used ReLU activation function except in the last layer where I have used the softmax activation. I have usually stuck to a learning rate between 0.01 and 0.001.

Varying the Activation Function

Each of the models contain 2 hidden layers of 100 neurons each, the regularization parameter has been set to zero for now and only accuracy over the training data is being checked. The entire training set is sent through the models for about 250 iterations. The learning rate has been chosen to maximize the training accuracy in each case.

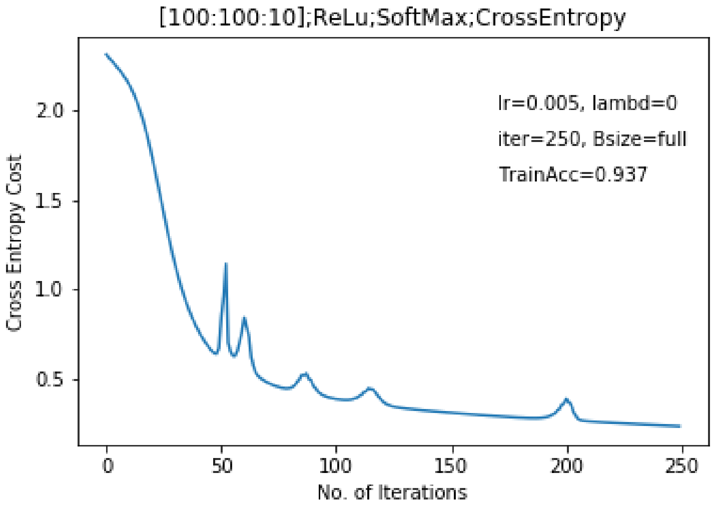


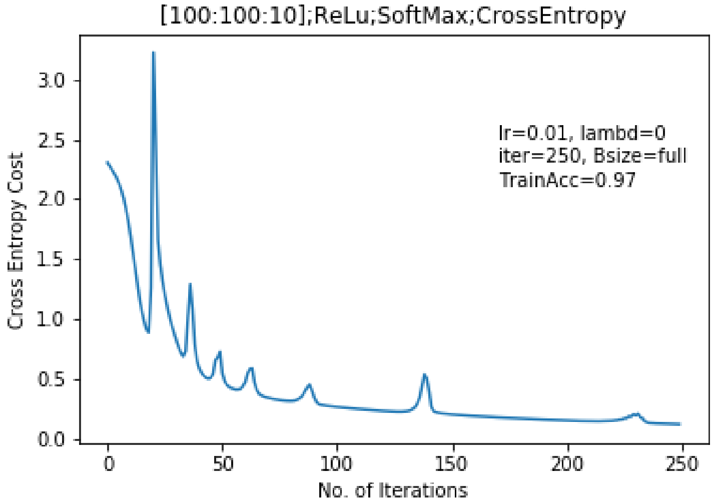


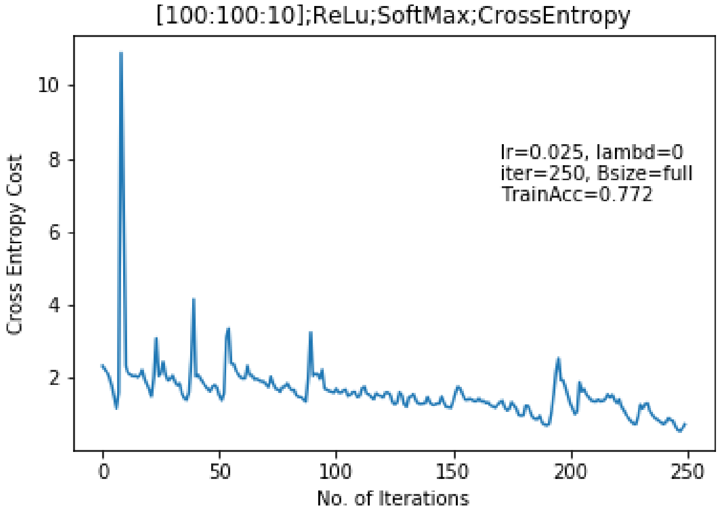


Varying the Learning Rate

The model with ReLU function has been used to see the effects of varying learning rate on. When the learning rate is low enough, sufficient accuracy is not reached on the training data itself for the given number of predetermined iterations, however when the learning rate is too high, our method fails to converge at all and gives even poorer accuracy.

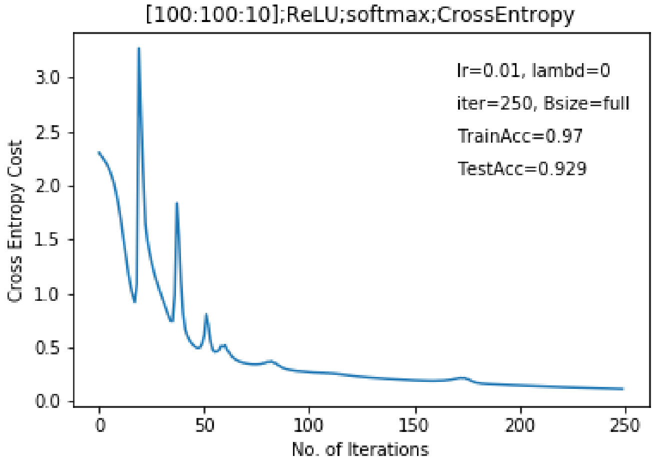


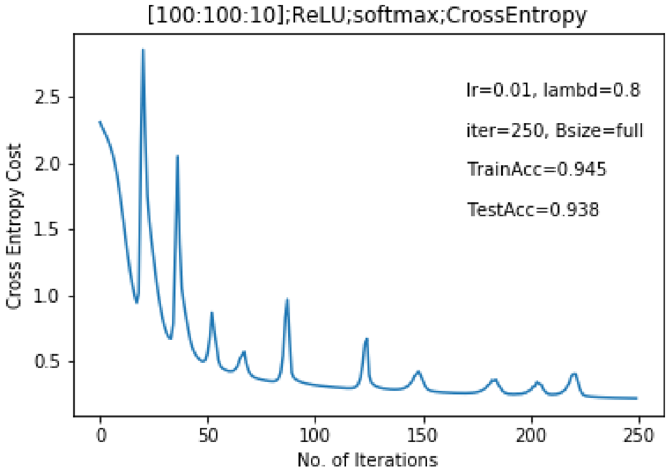


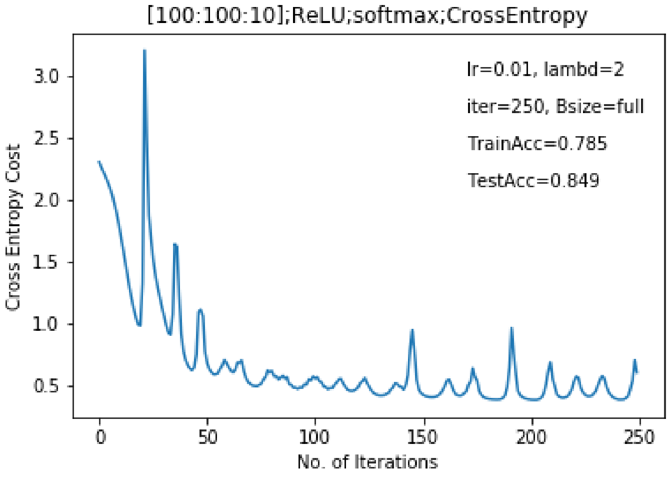


Varying the Regularization Parameter

The model with ReLU function has been used to see the effects of varying the regularization parameter on. When the regularization parameter is too less or zero, we run the risk of our model overfitting the data. This is seen in the first graph where our training accuracy exceeds the test accuracy by a huge margin. On the other hand, if the parameter is set too high, our model is not able to learn sufficiently from the data and performs poorly on both the data sets.

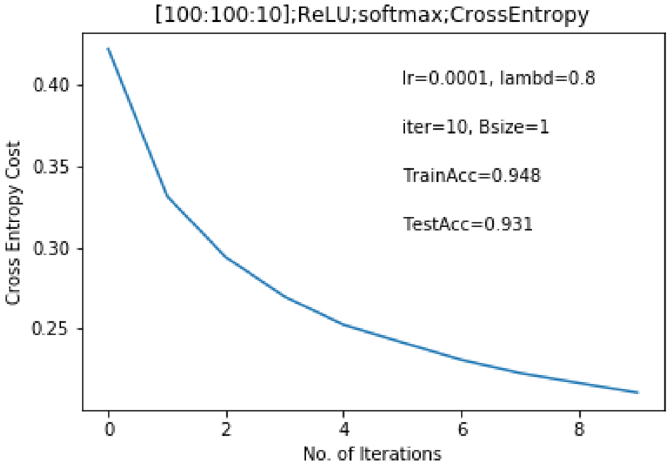


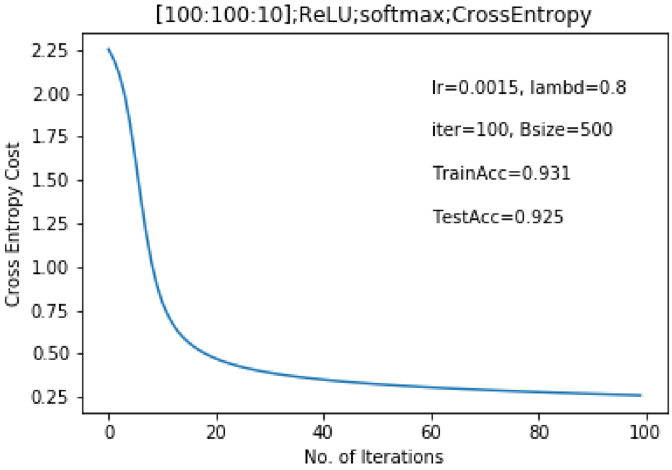


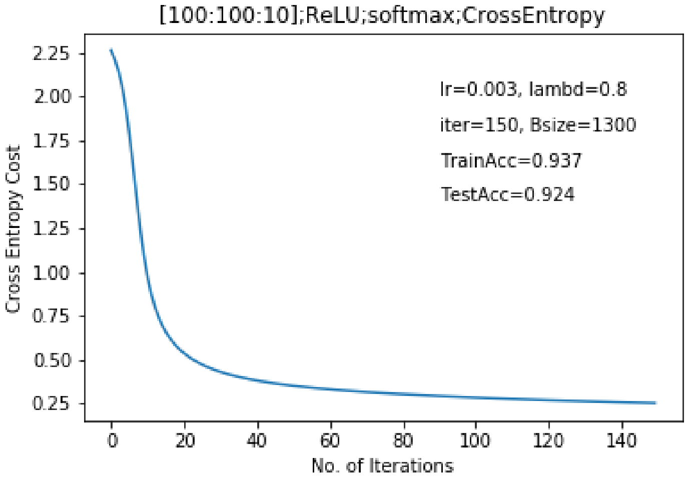


Varying the Batch Size

The model with ReLU function has been used to see the effects of varying the batch size on. The first graph is that of batch size 1 i.e. we have used stochastic error here instead of the actual error. Similarly the other two graphs are for batch sizes of 500 and 1300 respectively. A test set of 500 data points have been taken to cross validate. One interesting thing I noticed is that for convergence, smaller batch sizes work with smaller learning rate and less number of epochs/iterations.

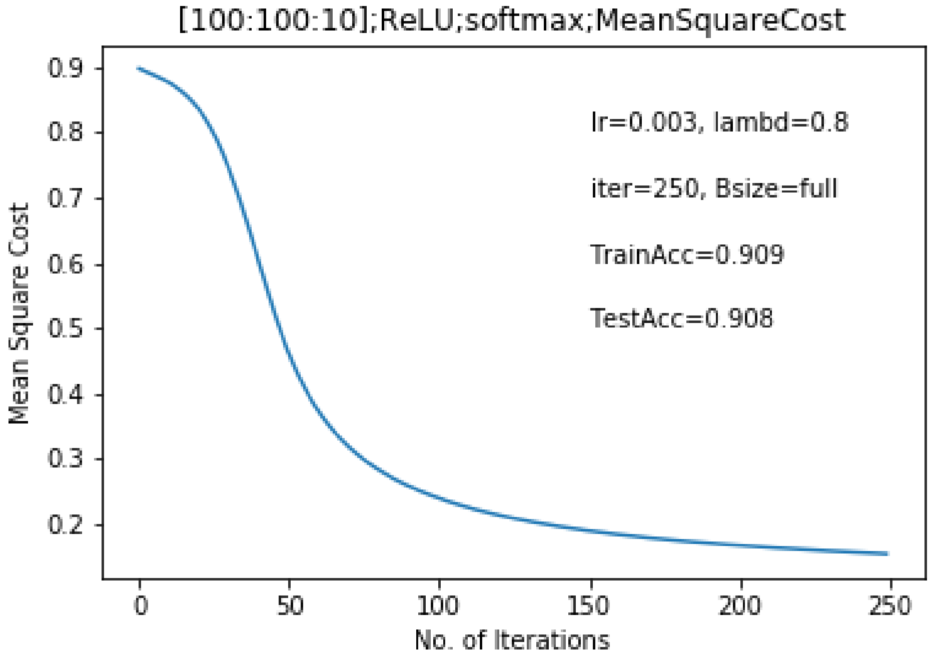


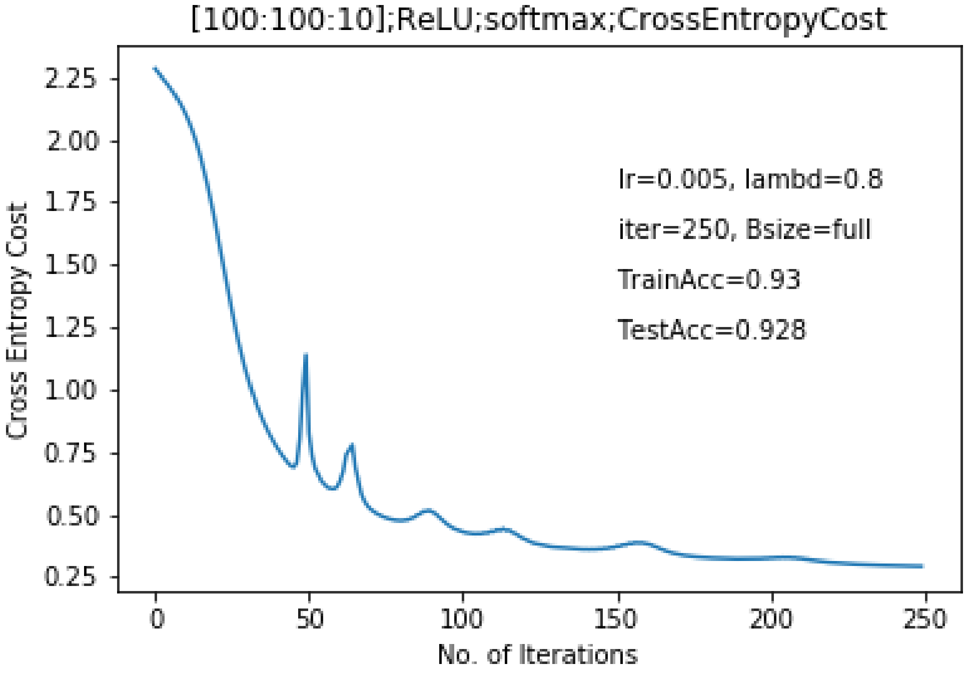




Varying the Activation Function

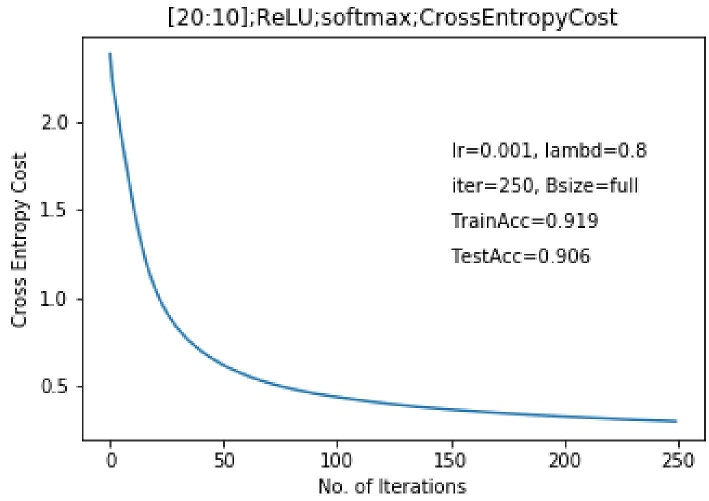
The model with ReLU function has been used to see the effects of varying the activation function on. As mentioned in the first page, cross entropy is considered to be a superior cost function for classification tasks, it is seen to outperform the model which uses Mean Squared Error in terms of both training and test data accuracy.

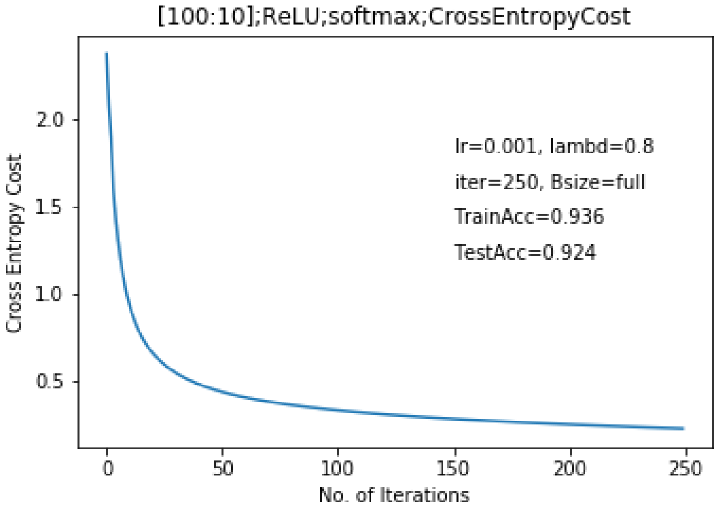


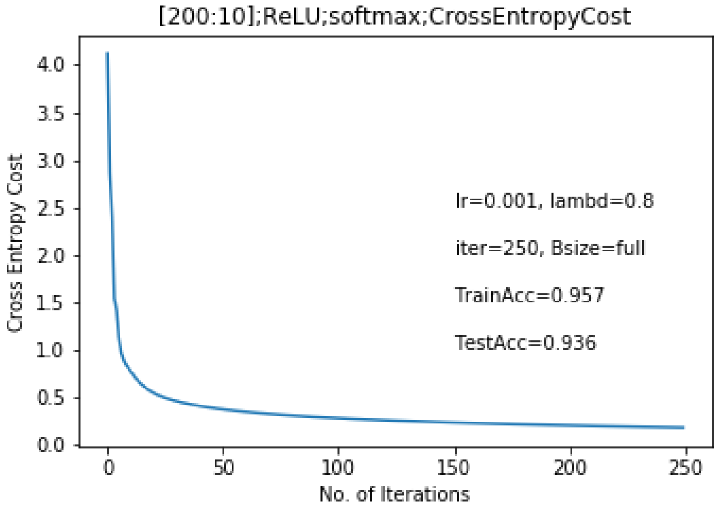


Varying the Number of Neurons

The model with ReLU function has been used to see the effects of varying the number of neurons on. For this I am choosing the model with only one hidden layer and varying the number of neurons from 20 to 100 to 200. As suspected, as the model capacity increases with number of neurons, it became easier to obtain a better training data accuracy for a fixed number of iterations.

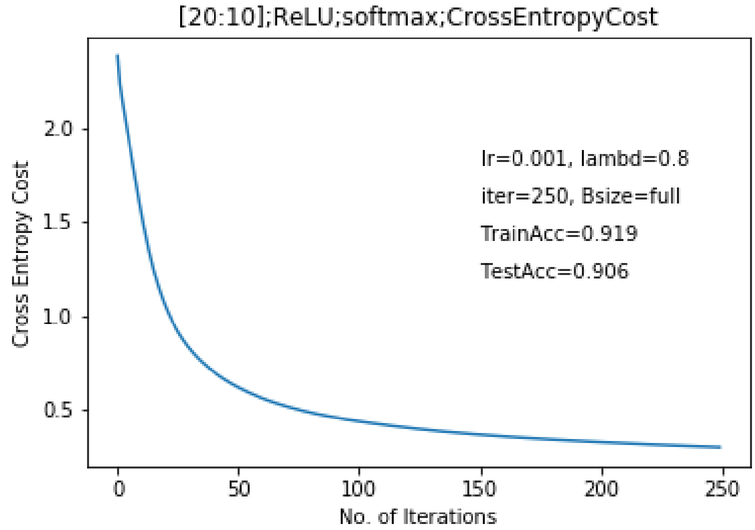


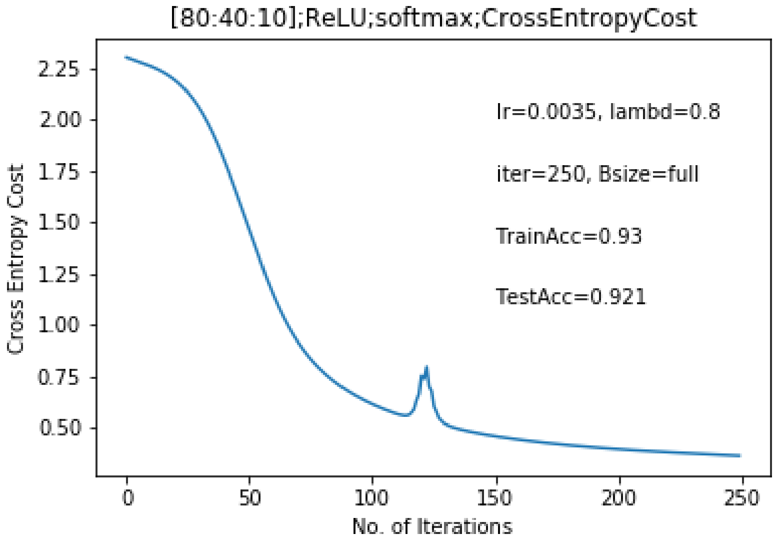


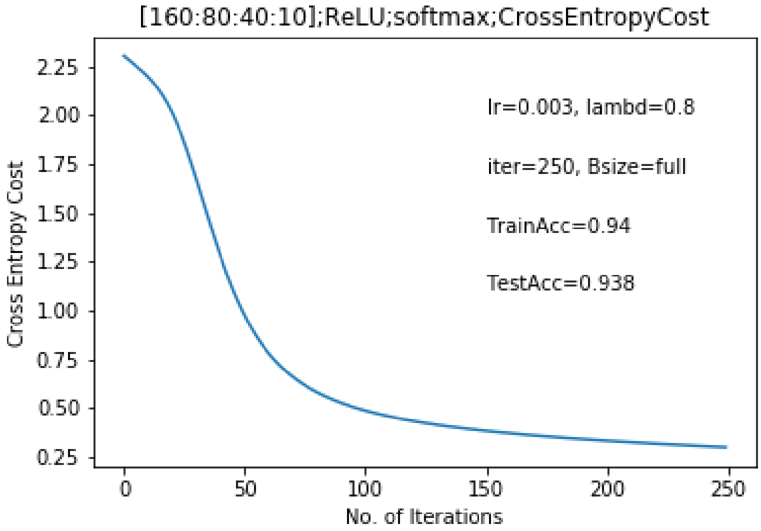


Varying the Number of Layers

The model with ReLU function has been used to see the effects of varying the number of layers on. Similar to what we saw in the previous page, increasing number of layers end up increasing the model complexity and hence better accuracy given the same number of epochs/iterations.


[20:10];ReLU;softmax;CrossEntropyCost
Cross Entropy Cost
No. of Iterations
lr=0.001, lambd=0.8
iter=250, Bsize=full
TrainAcc=0.919
TestAcc=0.906


[80:40:10];ReLU;softmax;CrossEntropyCost
Cross Entropy Cost
No. of Iterations
lr=0.0035, lambd=0.8
iter=250, Bsize=full
TrainAcc=0.93
TestAcc=0.921


[160:80:40:10];ReLU;softmax;CrossEntropyCost
Cross Entropy Cost
No. of Iterations
lr=0.003, lambd=0.8
iter=250, Bsize=full
TrainAcc=0.94
TestAcc=0.938