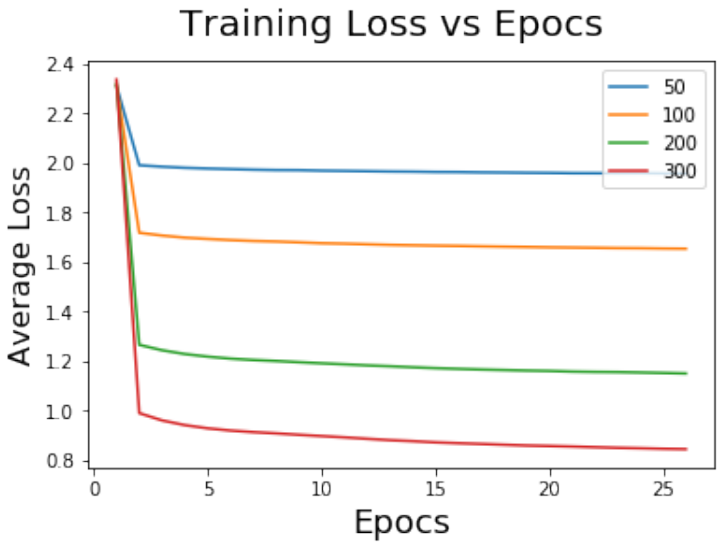
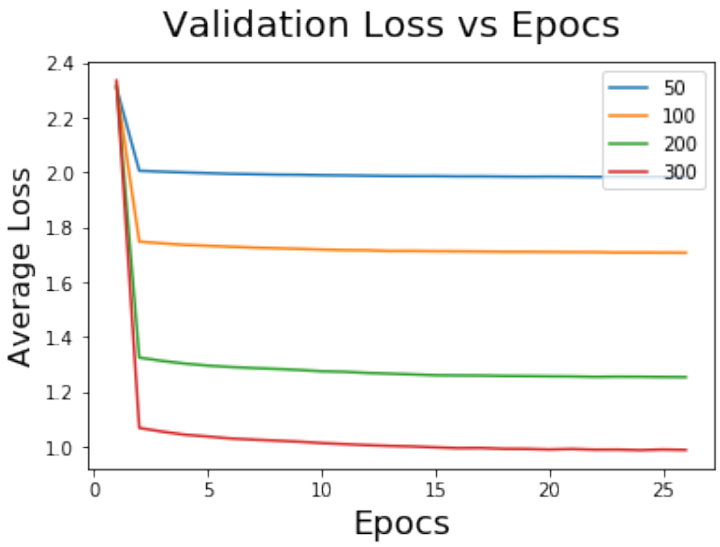
Question 1: Varying the size of the hidden layer (50, 100, 200, 300) - [keeping just one hidden layer] plot adam curves on training and validation curve

Solution:

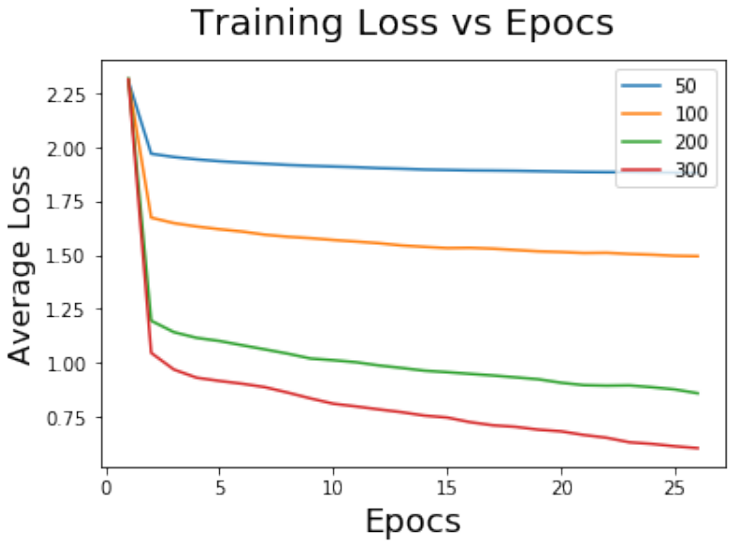


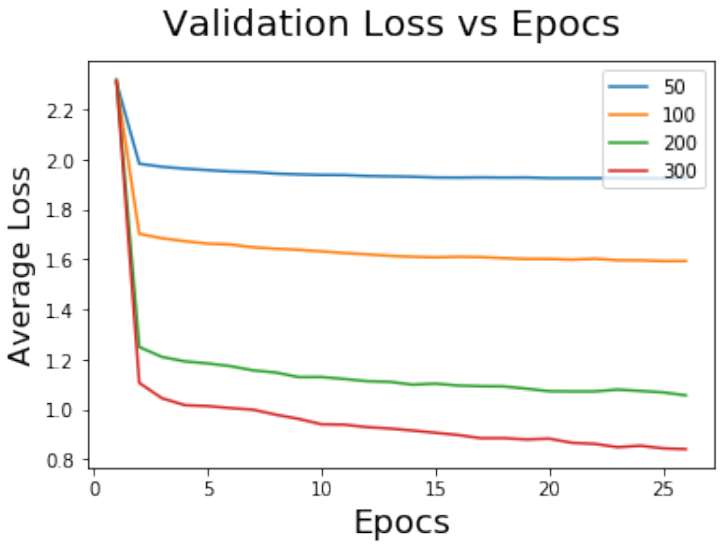


It can be concluded that as number of neurons are decreasing, final loss is increasing. This can be explained as more the number of neurons more will be complexity and hence it can learn better.

Question 2: varying the size of the hidden layer (50, 100, 200, 300) - [with two hidden layers and the same size for each hidden layer] plot adam curves on training and validation curve

Solution:

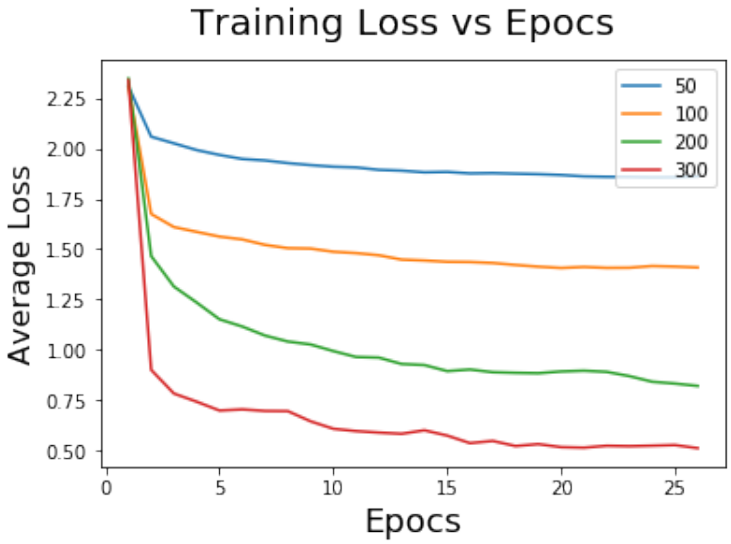


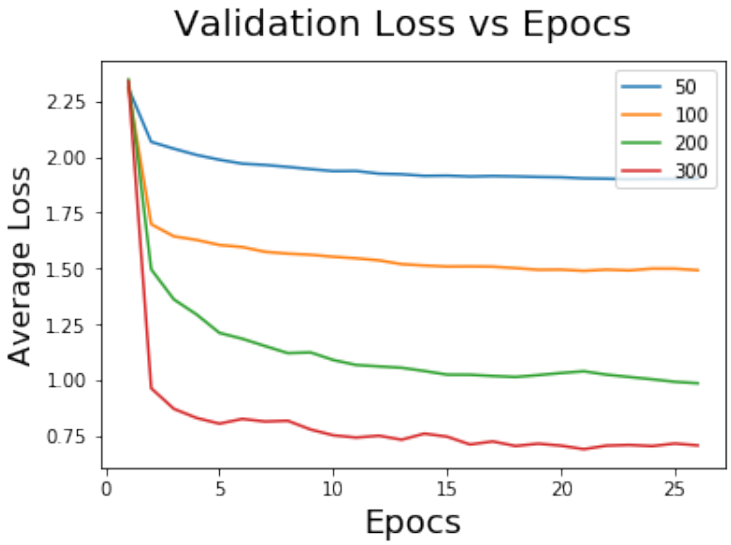


It can be concluded that as number of neurons are decreasing, final loss is increasing. This can be explained as more the number of neurons more will be complexity and hence it can learn better.

Question 3: varying the size of the hidden layer (50, 100, 200, 300) - [with three hidden layers and the same size for each hidden layer]. Plot adam curves on training and validation curve

Solution:

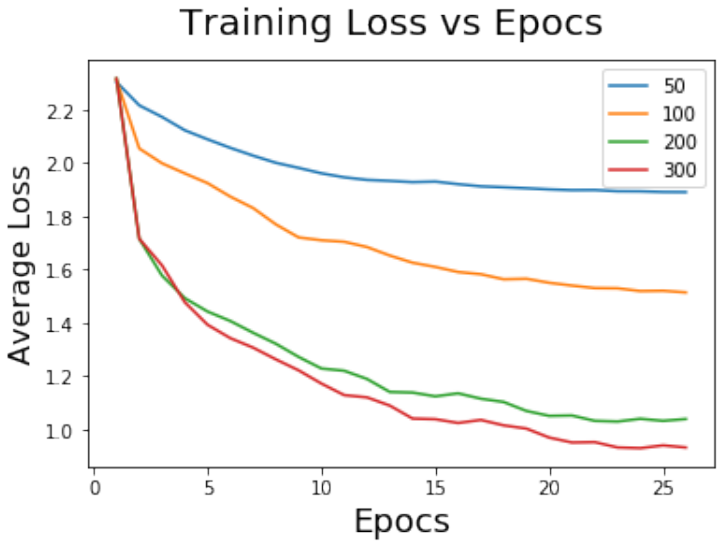


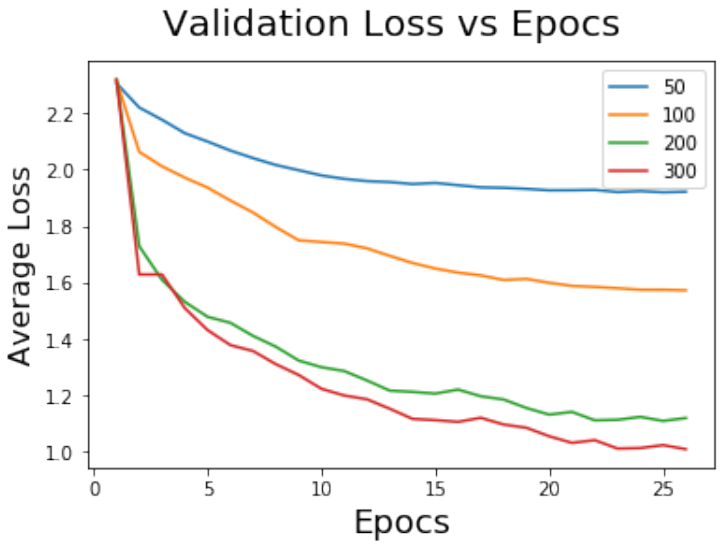


It can be concluded that as number of neurons are decreasing, final loss is increasing. This can be explained as more the number of neurons more will be complexity and hence it can learn better.

Question 4: varying the size of the hidden layer (50, 100, 200, 300) - [with four hidden layers and the same size for each hidden layer]. Plot adam curves on training and validation curve

Solution:

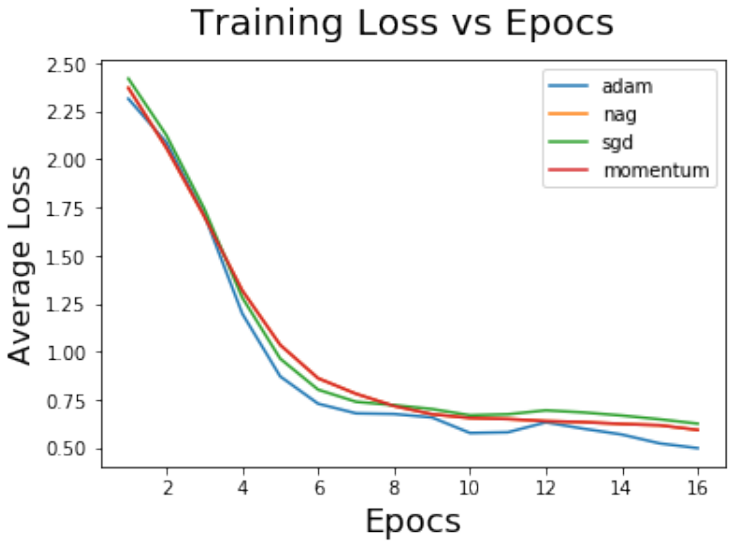


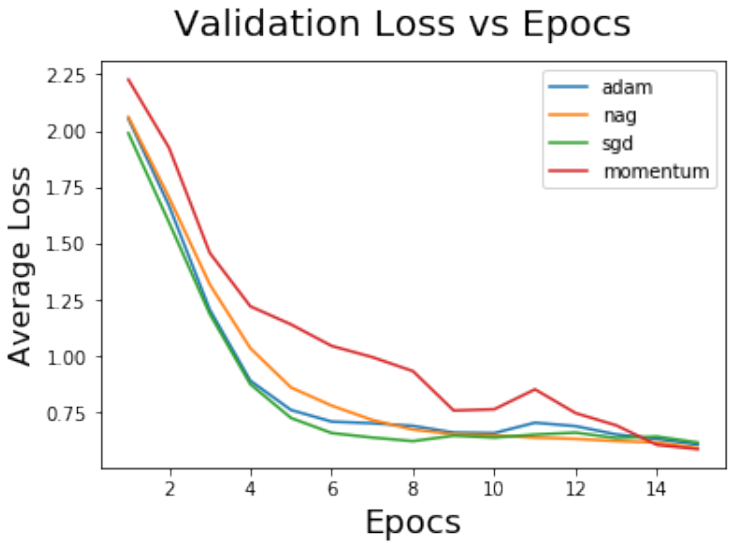


It can be concluded that as number of neurons are decreasing, final loss is increasing. This can be explained as more the number of neurons more will be complexity and hence it can learn better.

Question 5: Adam, NAG, Momentum, GD [with 4 hidden layers and each layer having 300 neurons] (again sigmoid activation, cross entropy loss, batch size 20 and tune the learning rate. momentum wherever applicable to get best results). You need to make the following plots:
x-axis: number of epochs, y-axis: training loss [4 curves - each curve corresponding to one of the four optimization methods mentioned above]
x-axis: number of epochs, y-axis: validation loss [4 curves - each curve corresponding to one of the four optimization methods mentioned above

Solution:

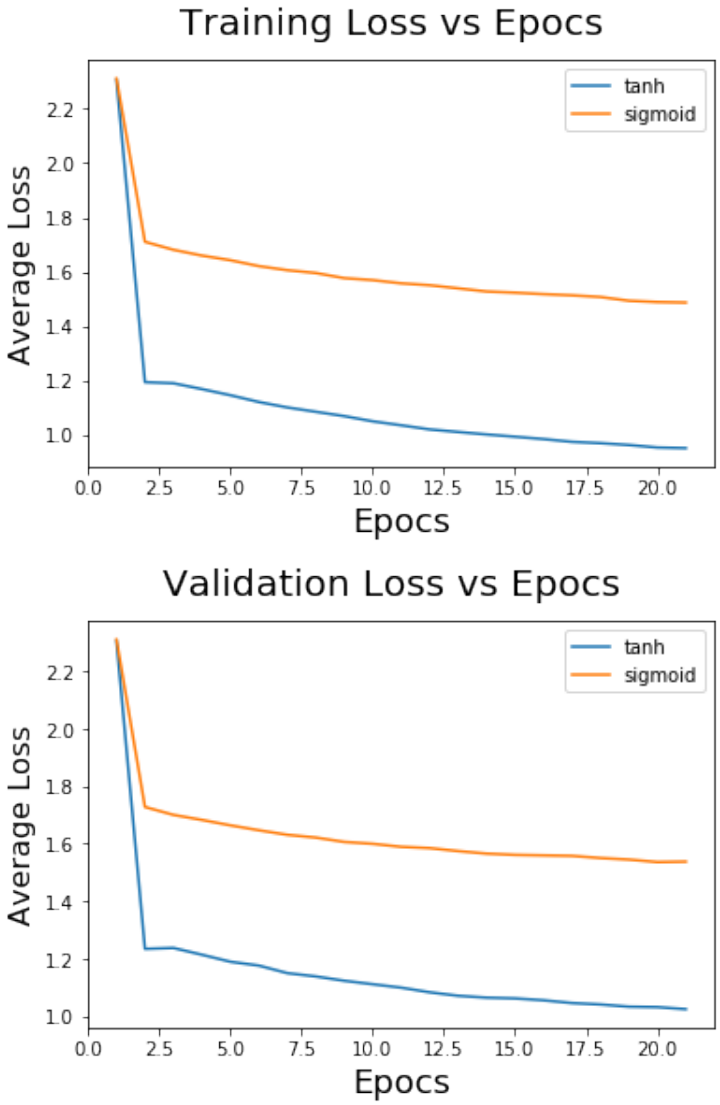




Momentum, nag , sgd and adam are performing similar when the network is very deep with many neurons. We have noticed that in validation, gardient descent performs very similar to how it performs in training. Adam is giving the lowest loss very fast, while momentum doesnot start well on validation but reaches to same convergence level.

Question 6: sigmoid v/s tanh activation [Adam, 2 hidden layers, 100 neurons in each layer, batch size 20, cross entropy loss]. You need to make the following plots:
x-axis: number of epochs, y-axis: training loss [2 curves - each curve corresponding to one of the two activation functions mentioned above]
x-axis: number of epochs, y-axis: validation loss [2 curves - each curve corresponding to one of the two activation functions mentioned above]

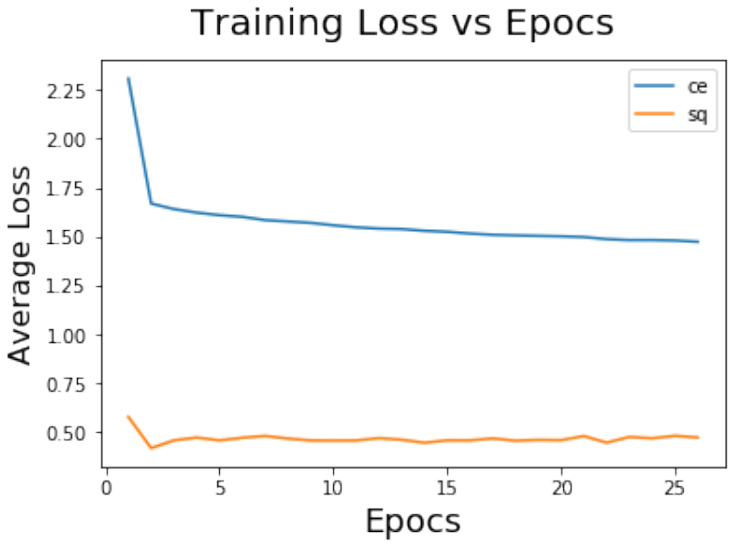
Solution:

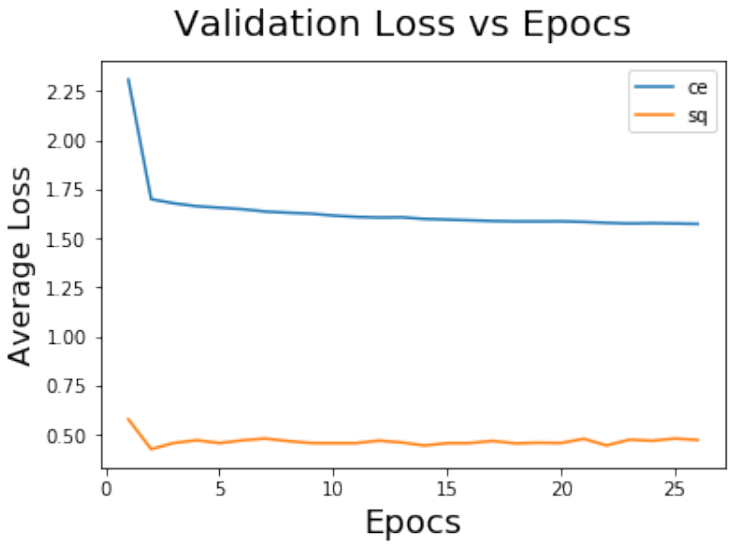


Tanh perfoms better than sigmoid in both validation and training dataset. The reason would be that sigmoid gives values between 0 and 1 but tanh give values between $\pi$and $-\pi$. Note: Both have been performed keeping same weight and learning rate initialization to have no bias of weights.

Question 7: cross entropy loss v/s squared error loss [Adam, 2 hidden layers, 100 neurons in each layer, batch size 20, sigmoid activation] You need to make the following plots:
x-axis: number of epochs, y-axis: training loss [2 curves - each curve corresponding to one of the two loss functions mentioned above]
x-axis: number of epochs, y-axis: validation loss [2 curves - each curve corresponding to one of the two loss functions mentioned above]

Solution:

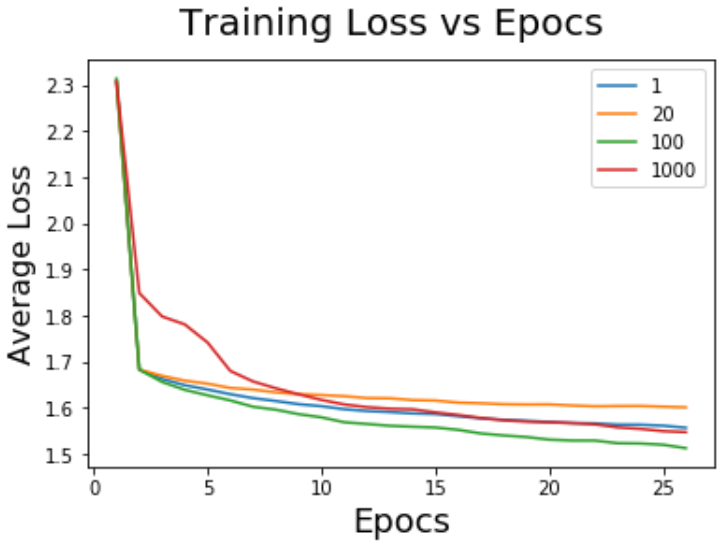


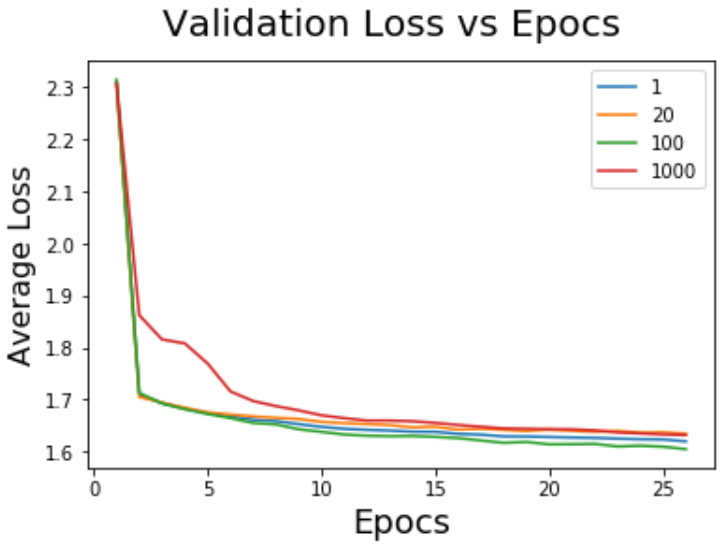


Square error is not able to explain the model properly as the output variable is integer and not real values. Cross entropy loss on other hand is working very well for the given data

Question 8: Batch size: 1,20,100,1000 [Adam, 2 hidden layers, 100 neurons in each layer, sigmoid activation, cross entropy loss] You need to make the following plots:
x-axis: number of epochs, y-axis: training loss [4 curves - each curve corresponding to one of the 4 batch sizes mentioned above]
x-axis: number of epochs, y-axis: validation loss [4 curves - each curve corresponding to one of the 4 batch sizes mentioned above]

Solution:



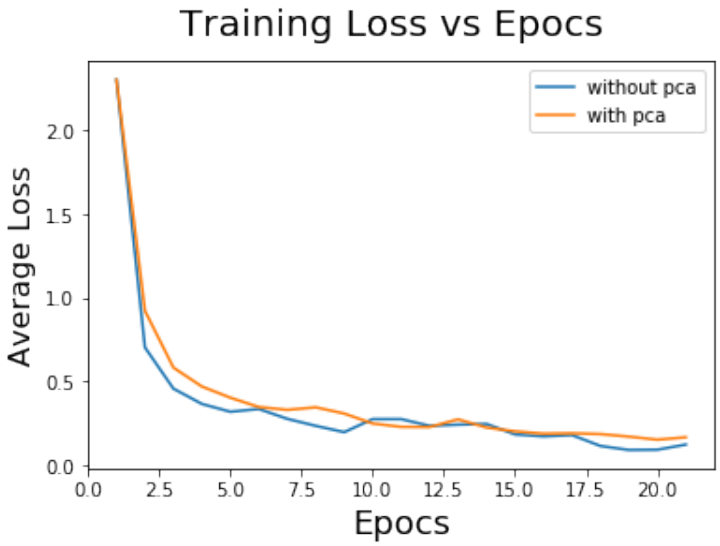


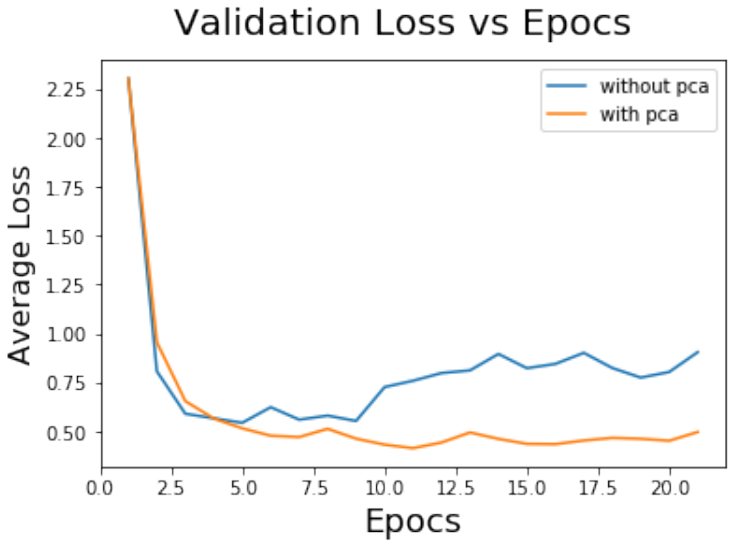
Batch size of 100 performs the best as it is the most optimum point. Batchsize of 1 will fluctuate the most and batch size of 1000 will have less updation of weights making it converge slower. Batch size 100 behave the best as it has optimum number of updation of weights.

Solution:

We also tried some extra things which helped us in attaing good accuracy.

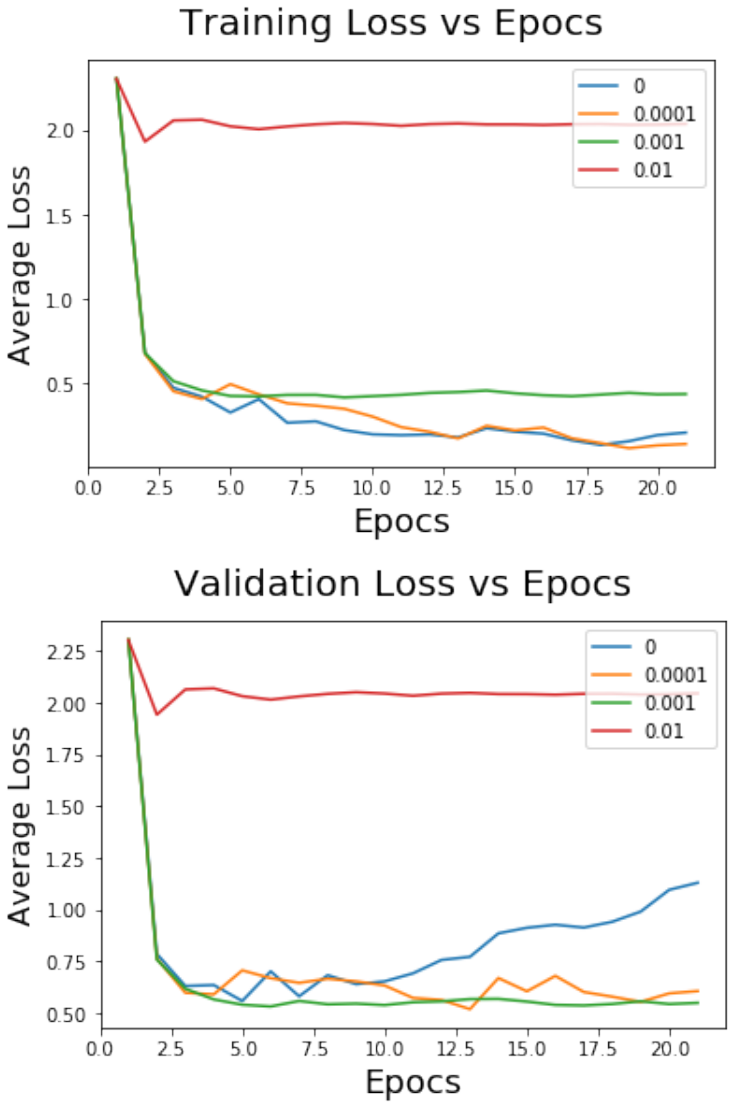
1. We tried using pca and choosing best 200 components.
2. We tried regularizing the loss using L2 regularization.
3. We also tried drop out technique to prevent overfitting in our model.
4. We tried relu activation function which performs very better that tanh or sigmoid activation function



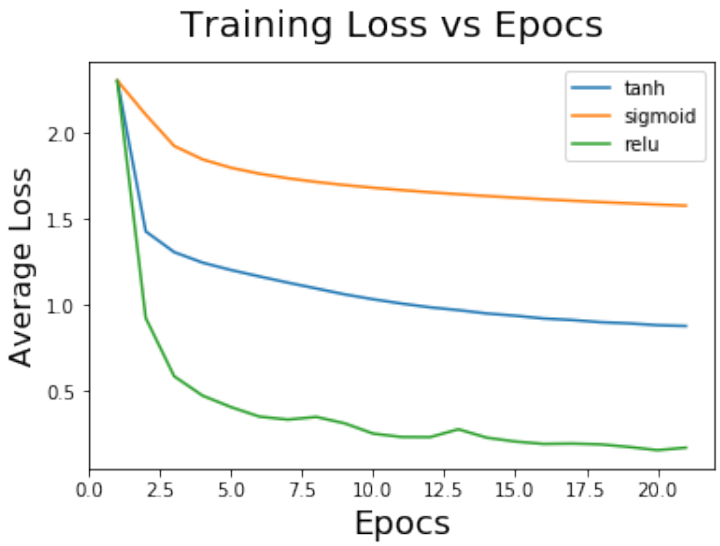


We can see that if we do not apply pca validation loss starts to increase as we increase epochs which means that after some epochs neural network starts to overfit and starts learning patterns which are noise.

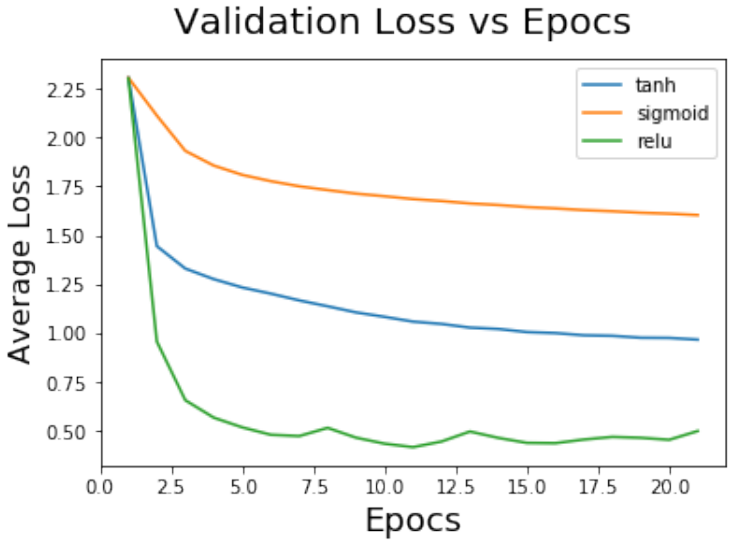
Solution:



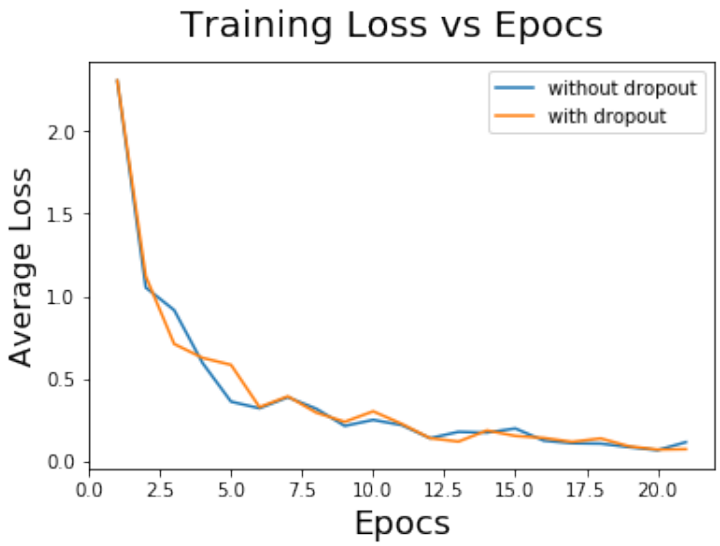
We can see that optimum value of regularizing parameter λ is 0.001. If λ is too high the model is unable to learn anything and if λ is zero the graph is unable to prevent overfitting.



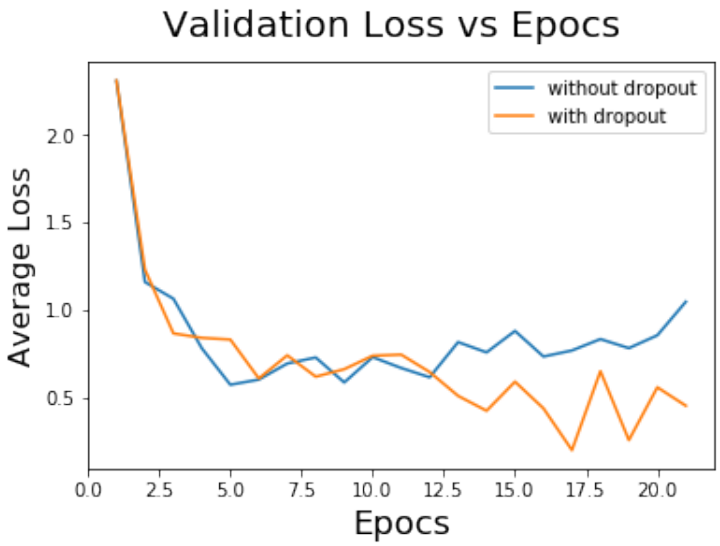
Solution:



Our accuracy on kaggle of 94.4 has been attained using relu. When we are using relu, convergence is extremely fast and loss on validation goes as low as 0.3.



Solution:



We can see that by adding drop out we have prevented overfitting.