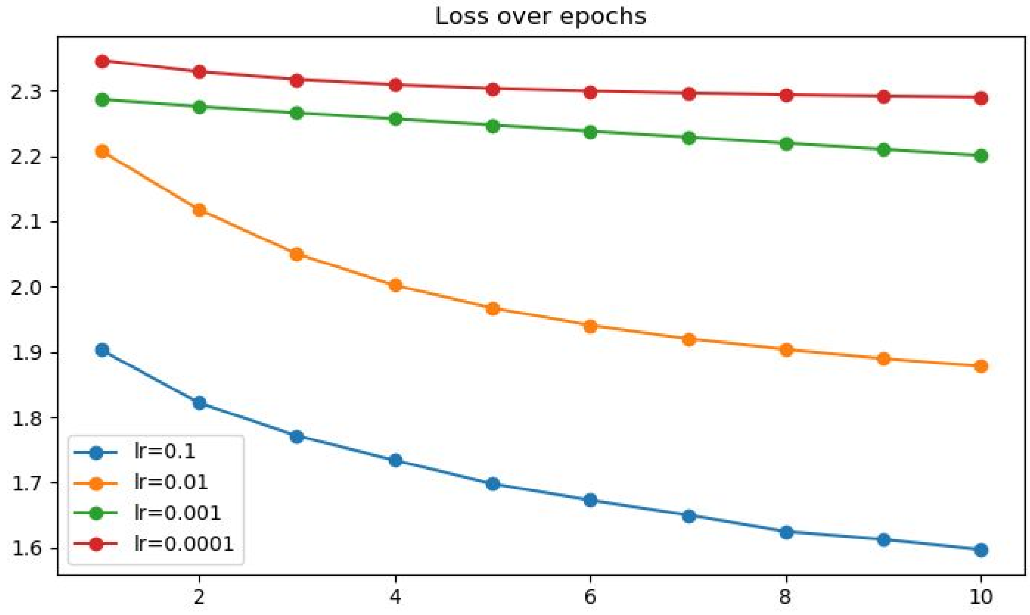
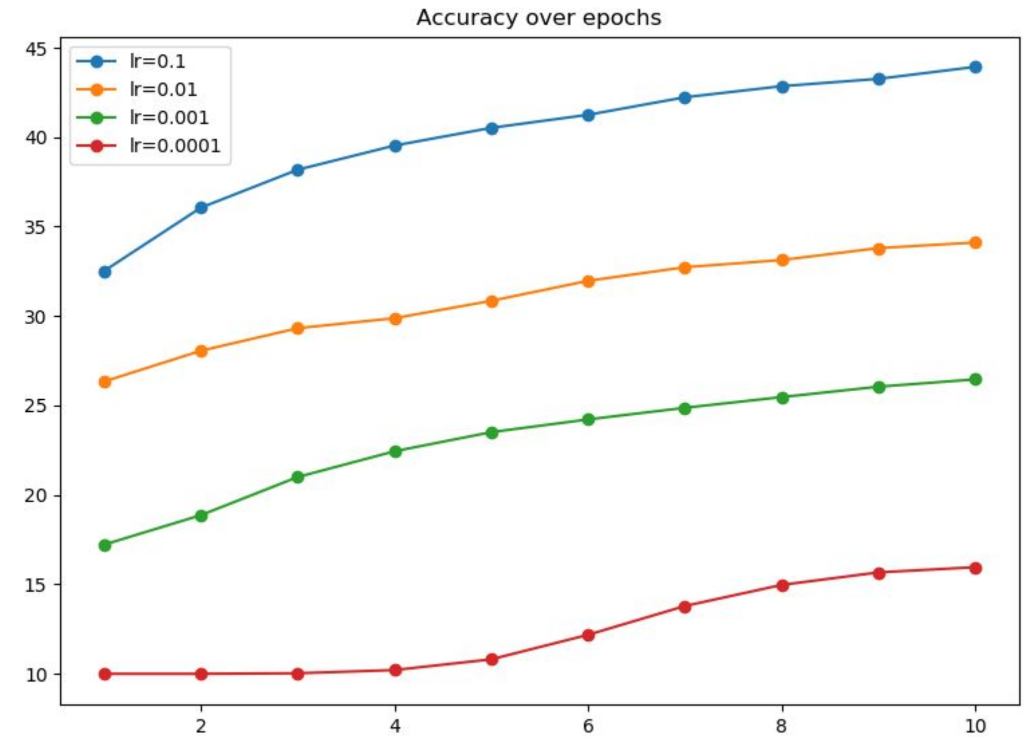
David Sahni, Chase McWhirt, Burton Jaursch
Implementation 3

Q1:

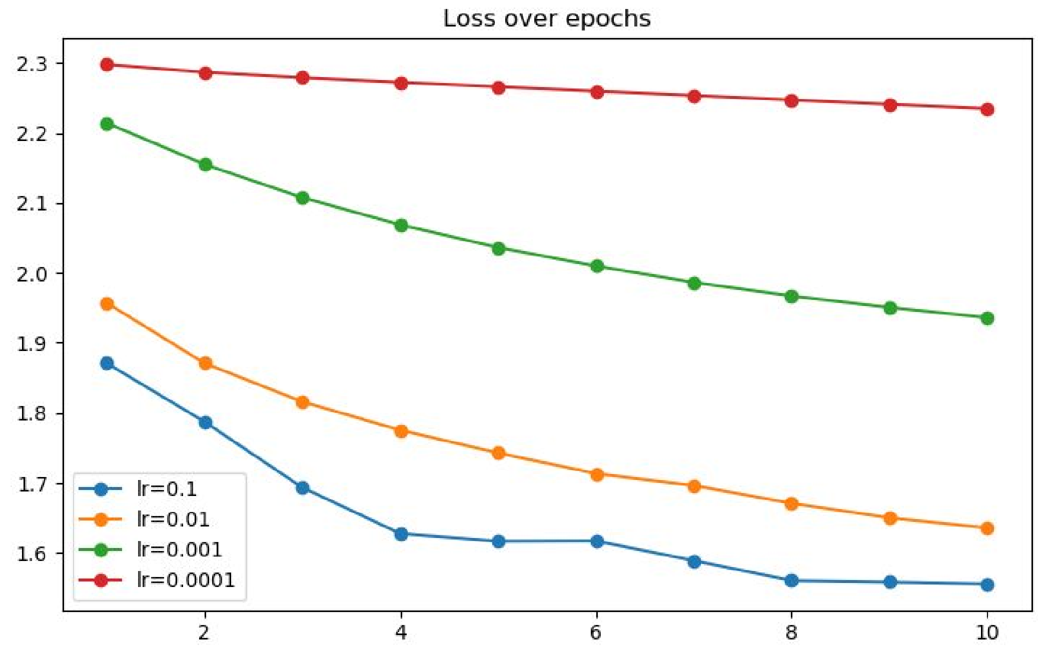
The charts below plot the loss, measured in negative-log-likelihood, and the validation accuracy over each epoch. Each line on the graph represents a different learning rate hyperparameter value, and these graphs indicate that a learning rate of .1 is optimal for this neural network. We can see it has the lowest loss and highest validation accuracy. We ended training at 10 epochs as loss and accuracy stopped significantly changing and runtimes were becoming long.

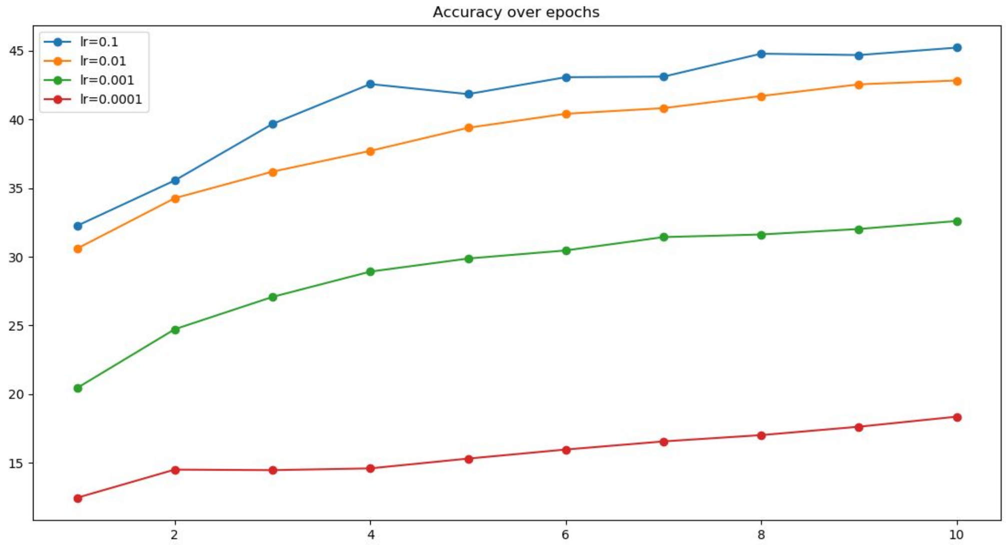




Best Results:
Test Accuracy: 43.93%
Learning Rate: .1

Q2:
Similar to the first question, these graphs represent the accuracy and loss for neural networks with different learning rates using the Relu activation function. Once again, we see the network with the highest learning rate be the most successful in terms of loss and validation accuracy. This may be due to the fact that this network is to "shallow" to properly learn the dataset, and thus the most aggressive learning rate provides the best results. Essentially a problem of underfitting. I ended training at 10 epochs as the loss changed slowed and to compare with the previous results.



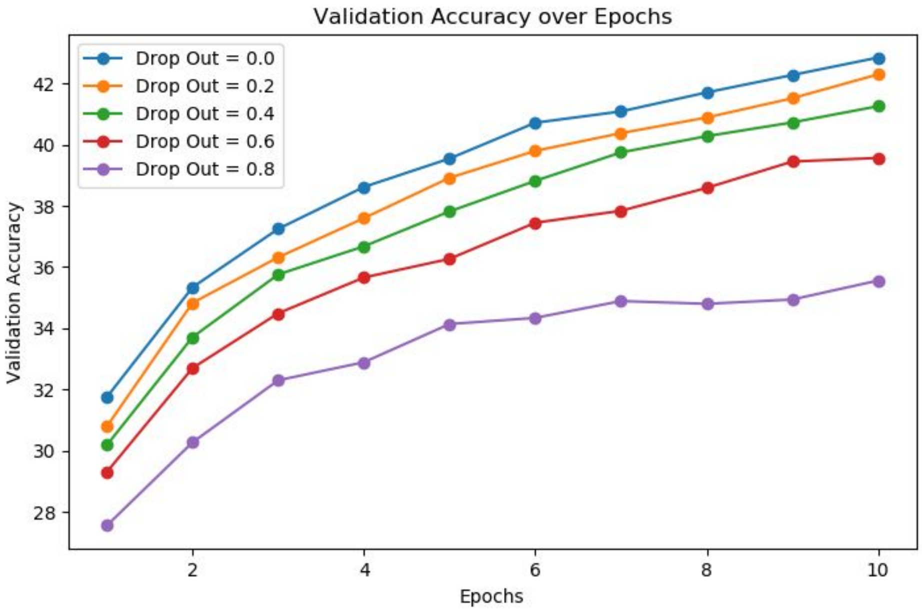


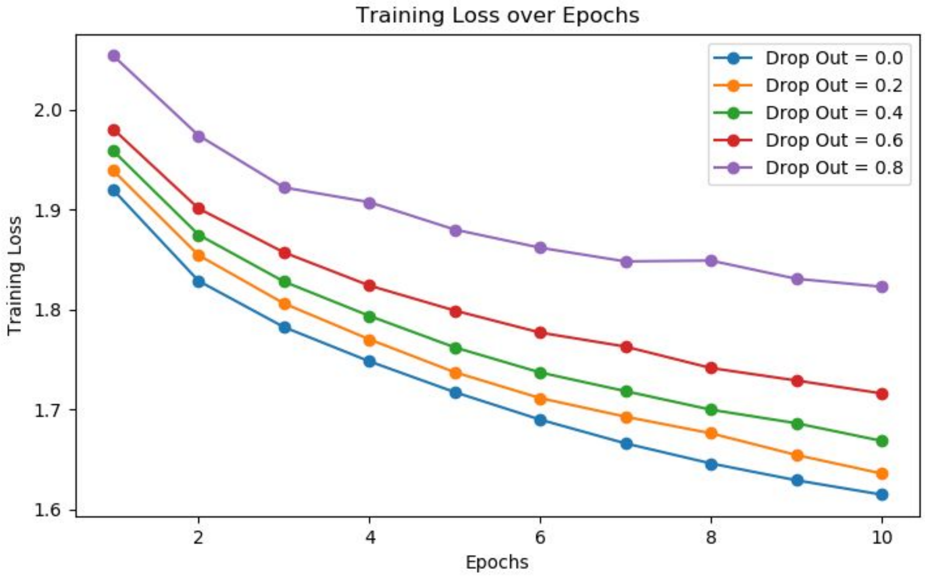
Best Test Accuracy: 45.2099%
Learning rate: .1

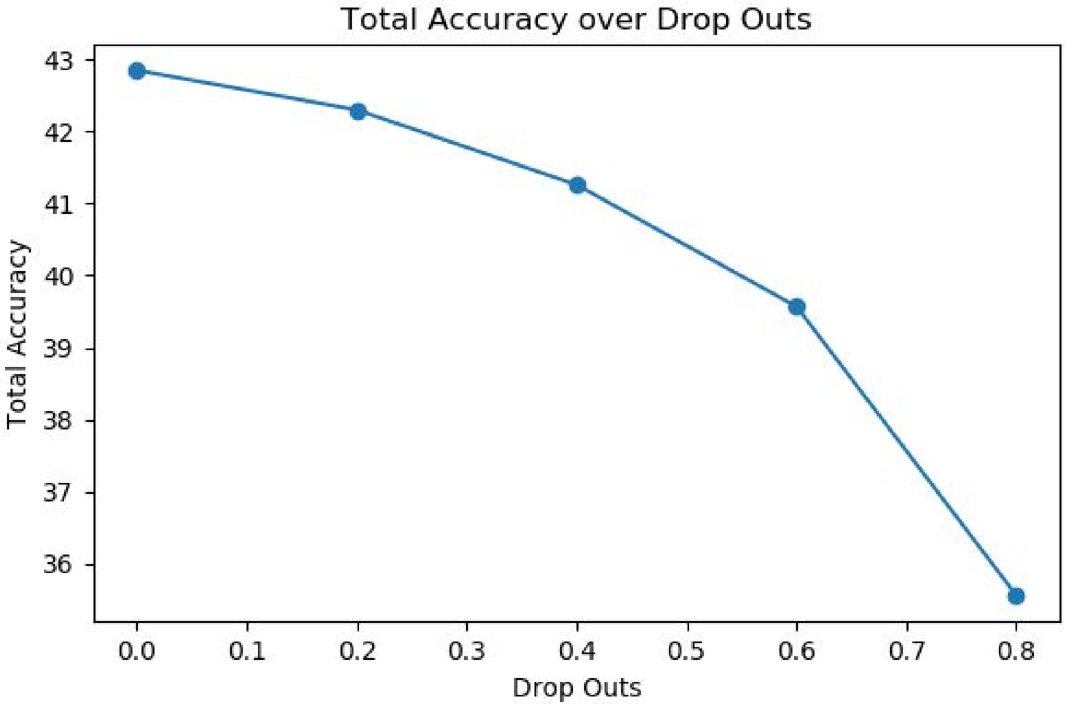
Q3:

Dropouts:

Dropouts is the concept of removing neurons from a neural network during the training phrase either during a forward or backward pass. This is generally done in order to prevent over-fitting. We are removing certain neurons to deal with the event that a neuron may not actually represent a good feature in the data, and to see how the removal of this neuron checks the accuracy of the data being displayed. Further, neurons will develop co-dependency to other neurons during training, which may also lead to overfitting and will need neurons to be seperated in order to remain accurate.



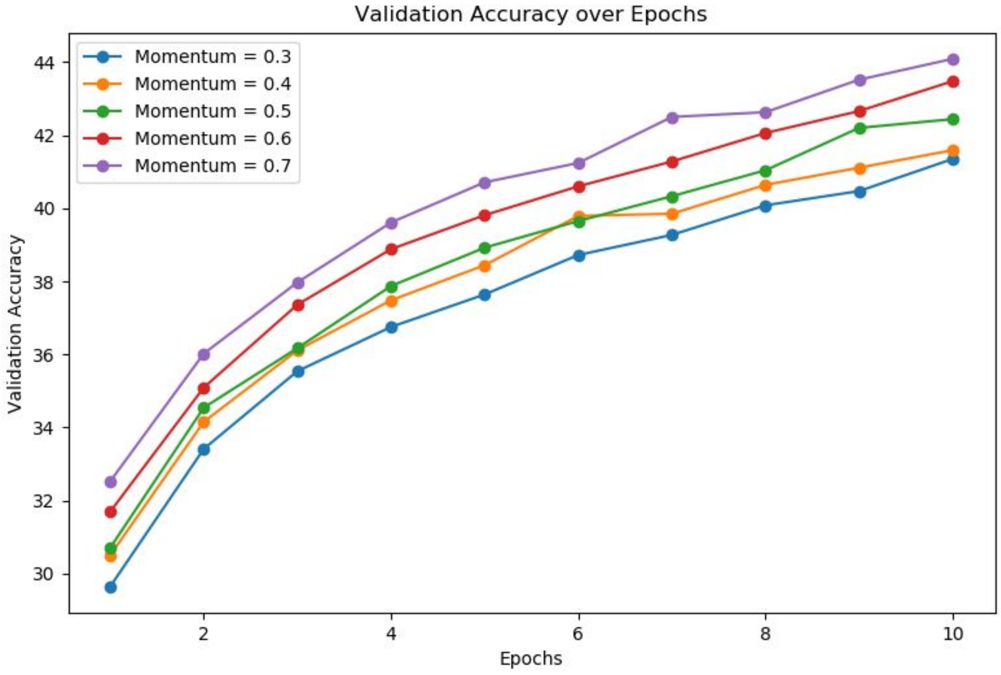


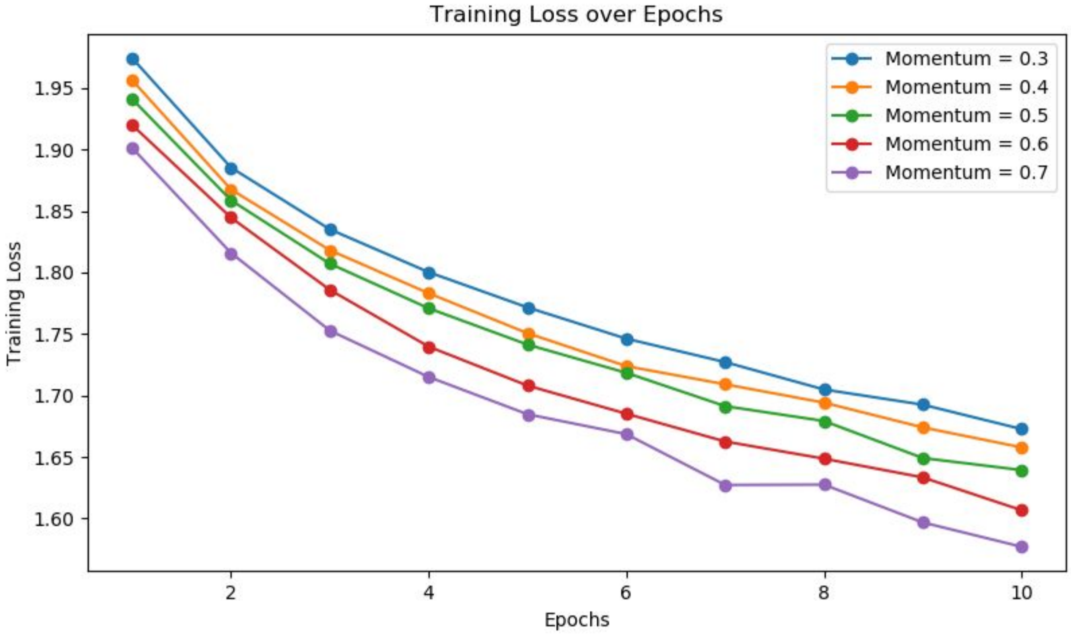


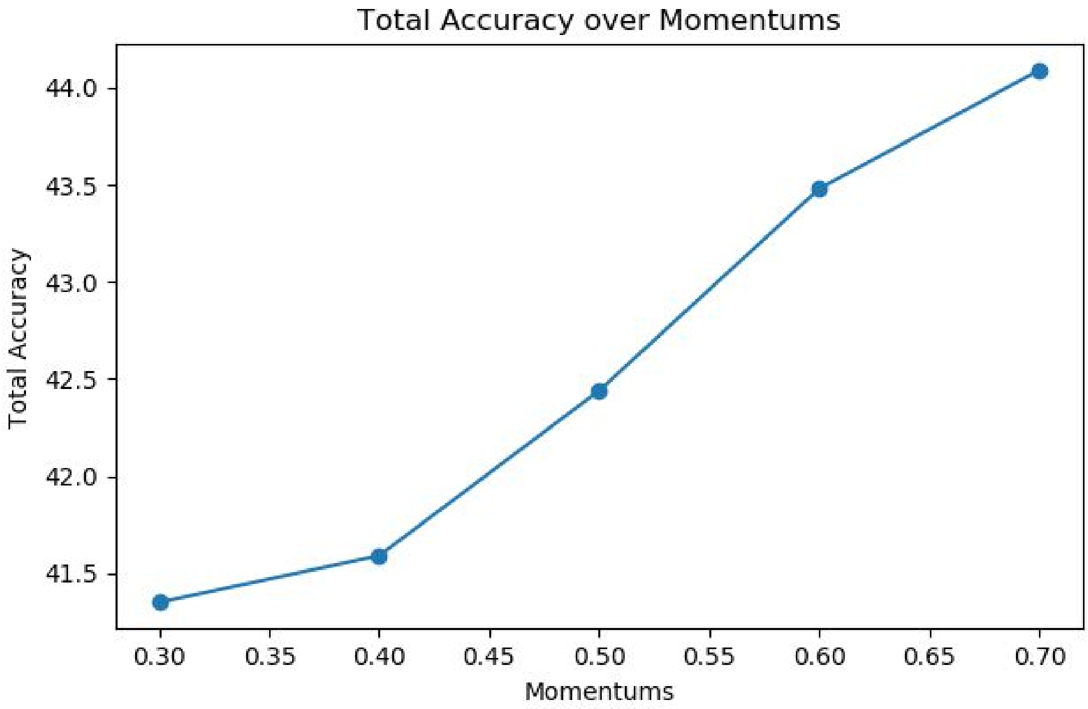
Through our testing, it is clear that convergence is faster when the dropout level is lower, and as dropout increases, convergence becomes slower. Dropout is a strategy to prevent overfitting, and since our model underfits the data, having a higher dropout negatively impacts our models performance. The total (testing) accuracy graph backs up that the optimal dropout should be 0 for the underfitted model.

Momentums:

Momentum is important for increasing the performance of training the model. It takes less attempts to converge towards a local maxima when momentum isn't too high. When momentum is too great, the model may never converge towards a local maxima as it can ignore gradient descent.


Validation Accuracy over Epochs


Training Loss over Epochs

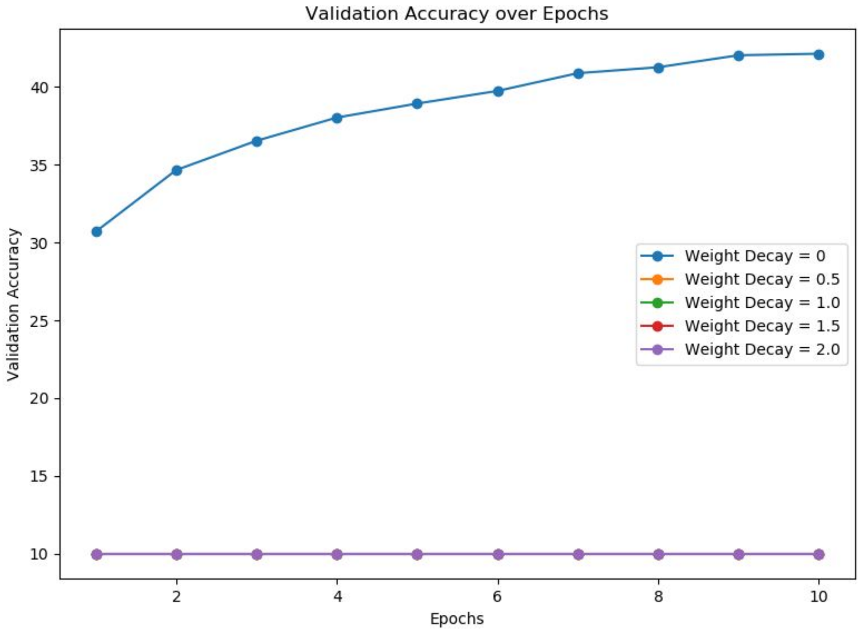


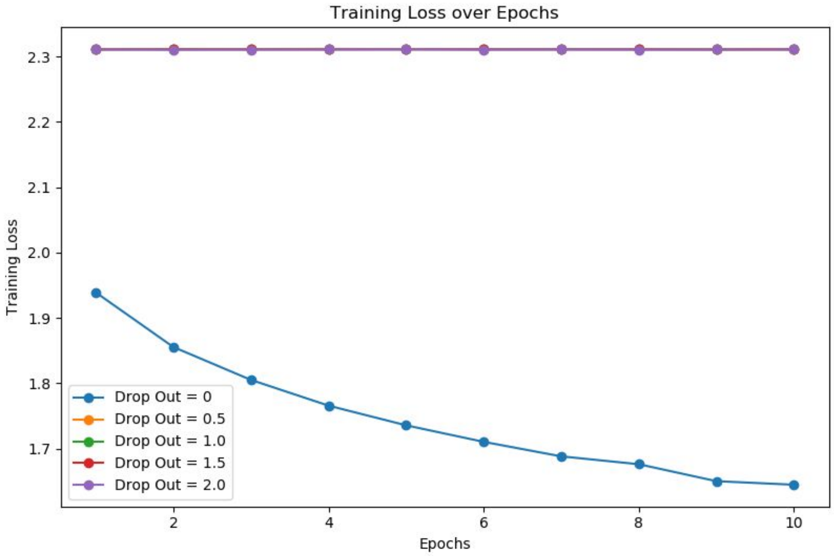
With the range of 0.3 to 0.7, momentum proved to be beneficial for training the model. As momentum increased, validation accuracy and total (testing) accuracy universally increased. Likewise, training loss universally decreased.

The range of tested momentums was actually greatly decreased (from 0.0-2.0 to 0.3-0.7) due to some erratic behavior. In the larger range, it appeared to completely ignore gradient descent, but in a bizarre way that wasn't clear from the data being evaluated. With the smaller range, it acts as predicted.

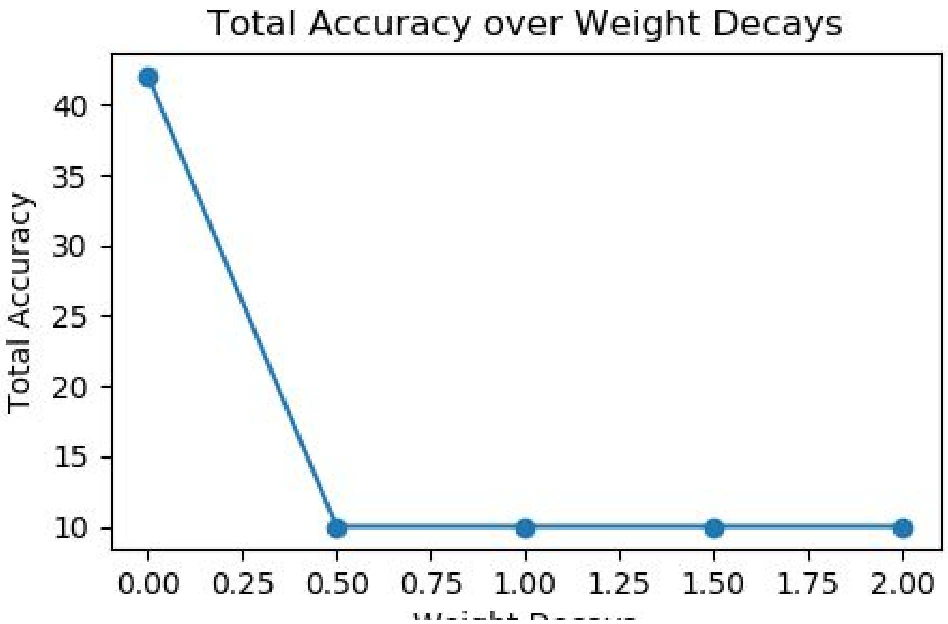
Weight Decays:

Weight decay is useful to prevent overfitting as the neural network fixates on a single feature. With weight decay, the final weights will be more generalized and thus increase the validation and testing accuracy, even if there is a small cost in training accuracy. However, if weight decay is too high, it will completely erode the learning efforts made by the neural network. In comparison to momentum, which, when high, causes the weights to overshoot local maxima, high weight decay causes weights to shift towards the origin.





Note: The legend in the above chart is incorrect, “Drop Out” should read “Weight Decay”

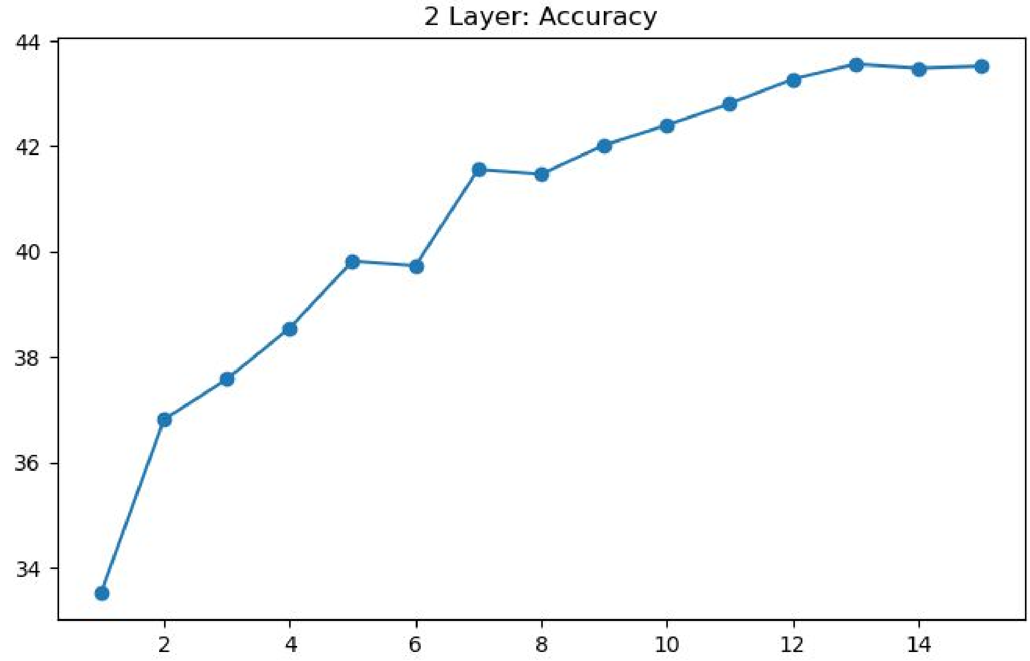


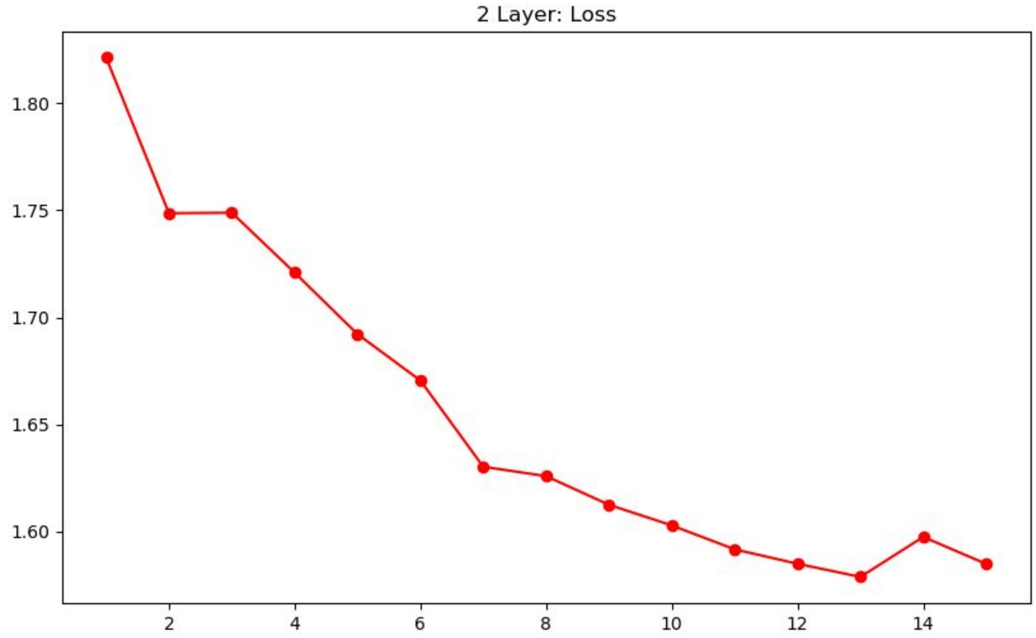
It is clear that a weight decay of 0.5 or higher results in extremely poor performance from the neural network. Accuracy is minimized and training loss is maximized. This is likely due to weight decay being much greater than the learning rate. Given more time, it'd be ideal to run with weight decay from 0-0.08 (0.00, 0.02, 0.04, 0.06, 0.08) since the learning rate is 0.1.

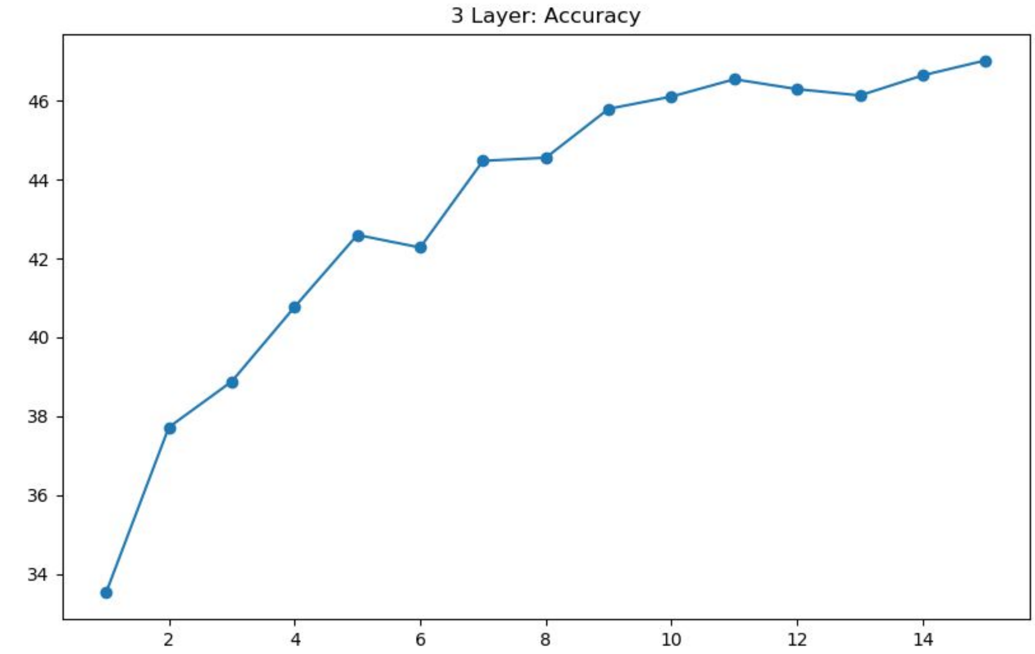
Q4:

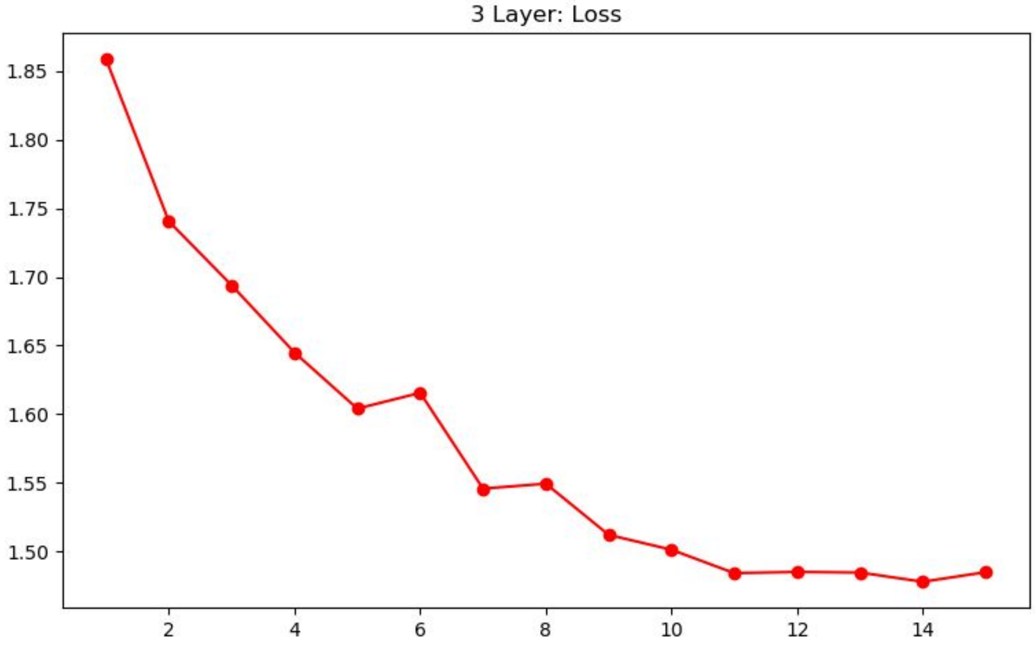
The two networks used the same hyperparameters: learning rate of 0.1, momentum of .5, and dropout/weight decay values of 0. I opted to use the Relu activation function as it provided slightly better results than sigmoid in q2.

The results shown below indicate that the 3-layer network (2 hidden layers of 50 nodes each) performed better than the 2-layer network (1 hidden layer of 100 nodes) in both testing and training. The two models ended with accuracies within 5% of each other in the mid 40s range, however the 3-layer network converged faster. This can be seen when comparing both the accuracy and the loss plots; the 2-layer network converges around epochs 12-13, where we see peaks in accuracy and minima in loss. The 3-layer network converges around epochs 10-11, where again we see max and min values for accuracy and loss respectively. This is likely due to 3-layered network having an additional layer on each backward pass allowing for more learning, essentially having a more complex model. Since our training accuracy is around 45%, it is likely that our model has an issue of underfitting, and therefore increasing the complexity of the network by adding more layers reduces this issue and raises accuracy.




2 Layer: Loss
1.80
1.75
1.70
1.65
1.60
2
4
6
8
10
12
14


3 Layer: Accuracy
46
44
42
40
38
36
34
2
4
6
8
10
12
14


3 Layer: Loss
1.85
1.80
1.75
1.70
1.65
1.60
1.55
1.50
2
4
6
8
10
12
14

.

	Training Accuracy	Testing Accuracy
2-Layer	45.125	43.520
3-Layer	48.970	47.030