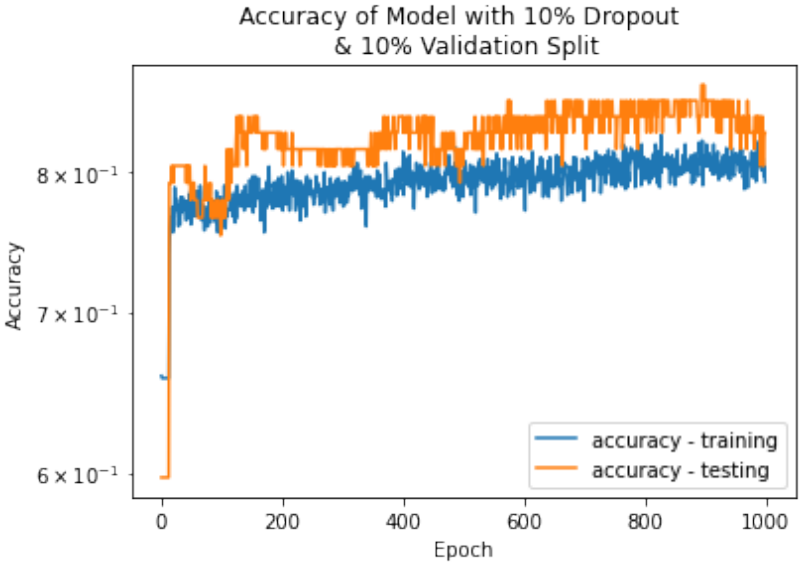
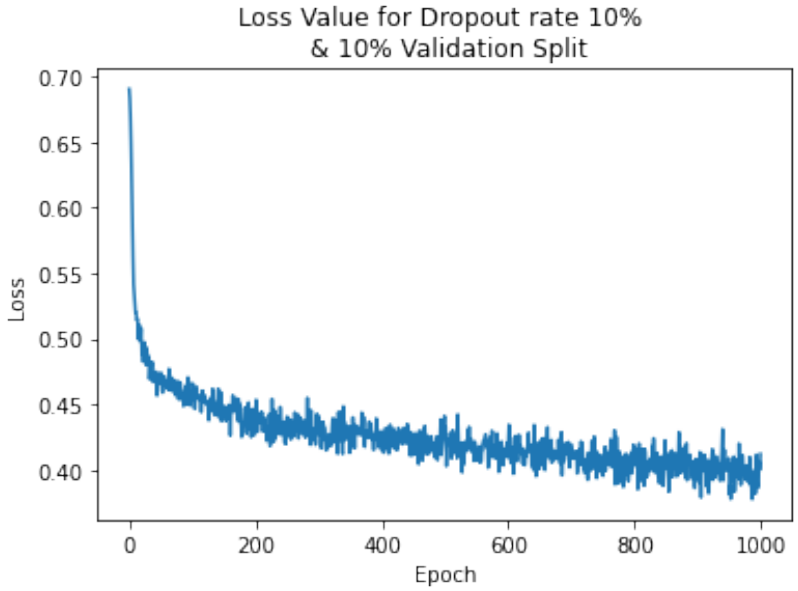
Steven Vasquez
04/25/2020
Professor Russ
Deep Learning

Dropout Rate/Validation Split Mini Report pt. 2

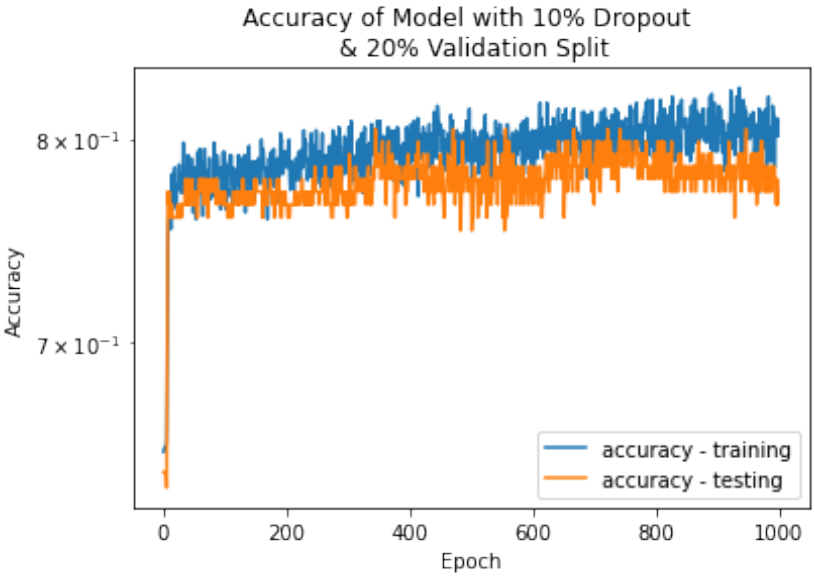
Dropout' is one way of preventing or accommodating a neural network that is overfitting. Besides 'dropout' adding data to a model can help decrease the overfitting. The dropout technique is used to 'drop' links, weights, and neurons to help simplify the model for its optimal weights and biases. For this example different combination of dropout rate and validation split effect the accuracy and confusion matrix of a binary classification neural network. A separate notebook will be attached to show the code that was used for this demonstration. The main differences in the commands are the dropout rate and validation split. For the first 3 examples the dropout rate will be 10% while the validation split will be 10%, 20%, and 30% in that order. This is the same as the first example but a crucial aspect is added to this code compared to the previous one. Checkpoints are added to this ANN to also prevent overfitting.

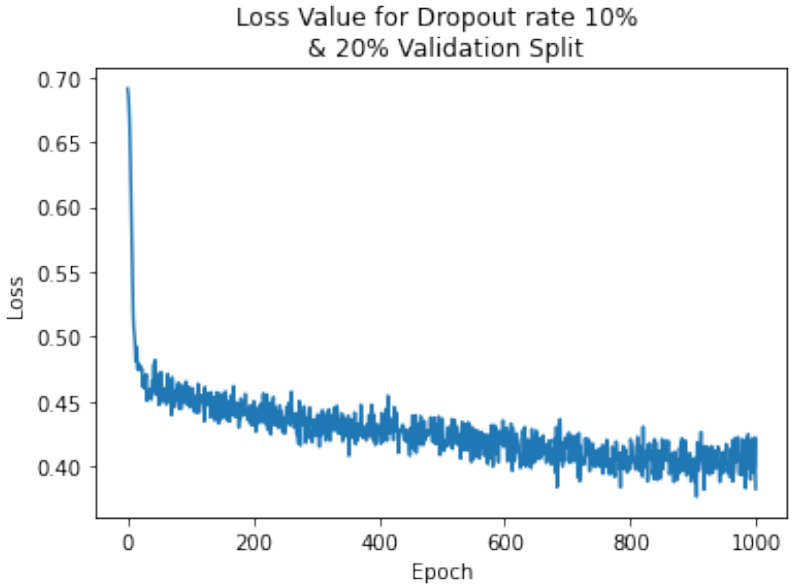
```
768/768 [==============================] - 0s 27us/step

accuracy: 83.33%
Confusion Matrix
================
True negatives:  448
False negatives:  76
False positives:  52
True positives:  192
```



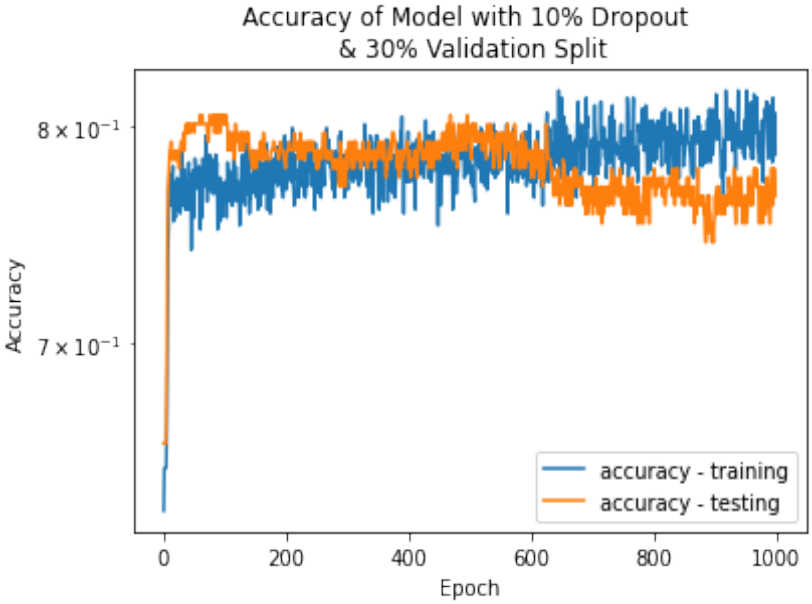


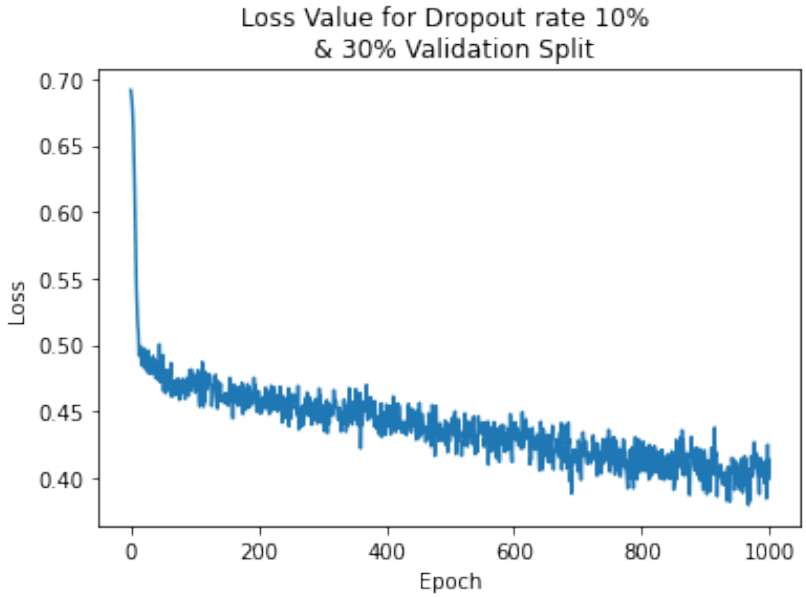
```
768/768 [==============================] - 0s 24us/step

accuracy: 82.03%
Confusion Matrix
================
True negatives:  447
False negatives:  85
False positives:  53
True positives:  183
```





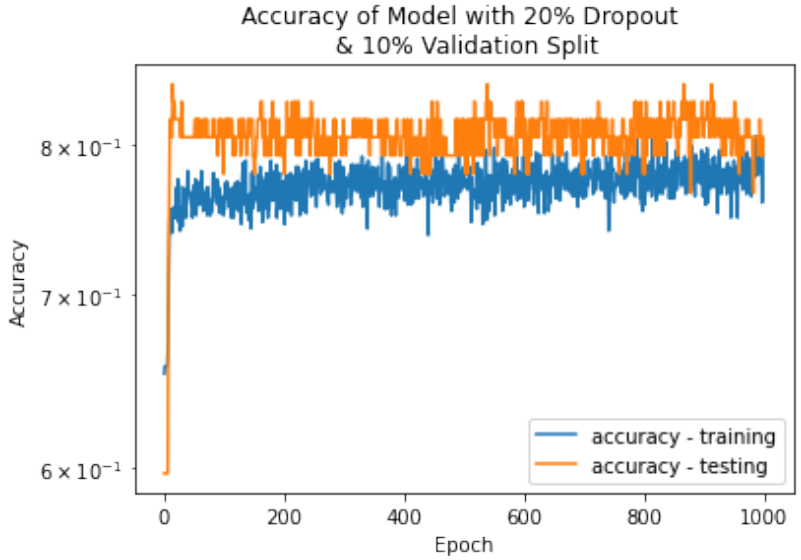
```
768/768 [==============================] - 0s 23us/step

accuracy: 80.73%
Confusion Matrix
================
True negatives:  436
False negatives:  84
False positives:  64
True positives:  184
```

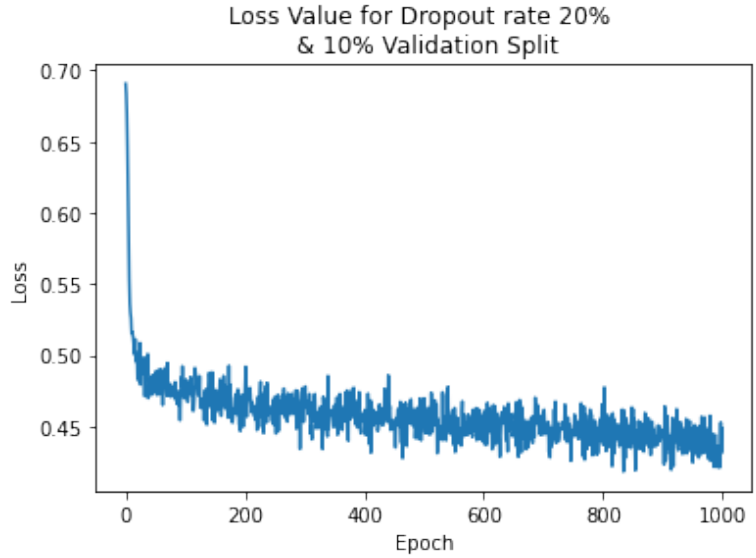




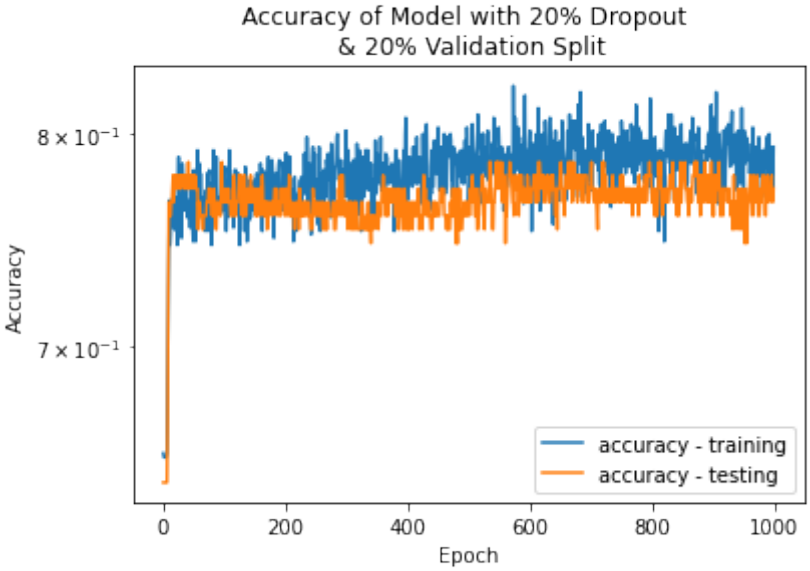
Now the same process will be done again but for 20% dropout rate.

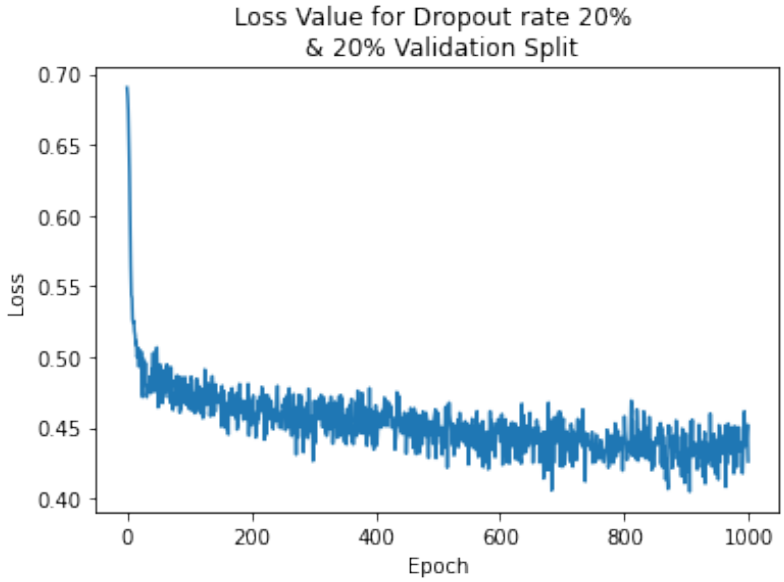
```
768/768 [==============================] - 0s 24us/step

accuracy: 80.99%
Confusion Matrix
================
True negatives:  442
False negatives:  88
False positives:  58
True positives:  180
```



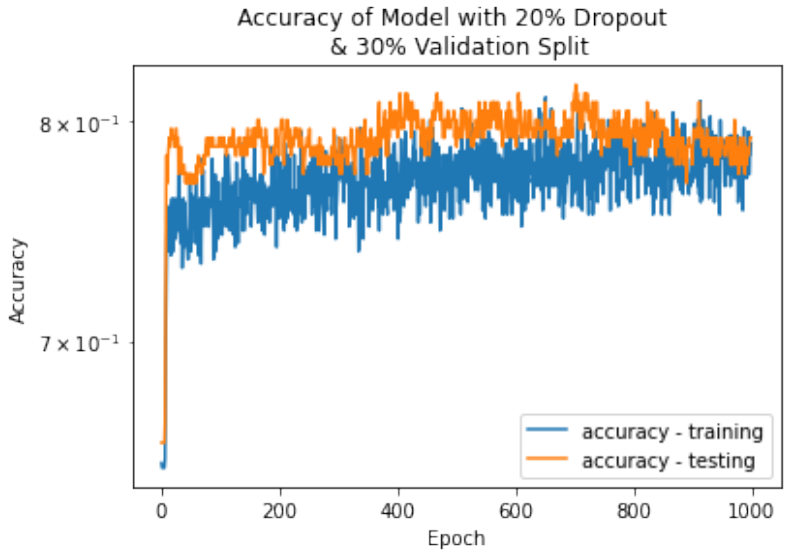


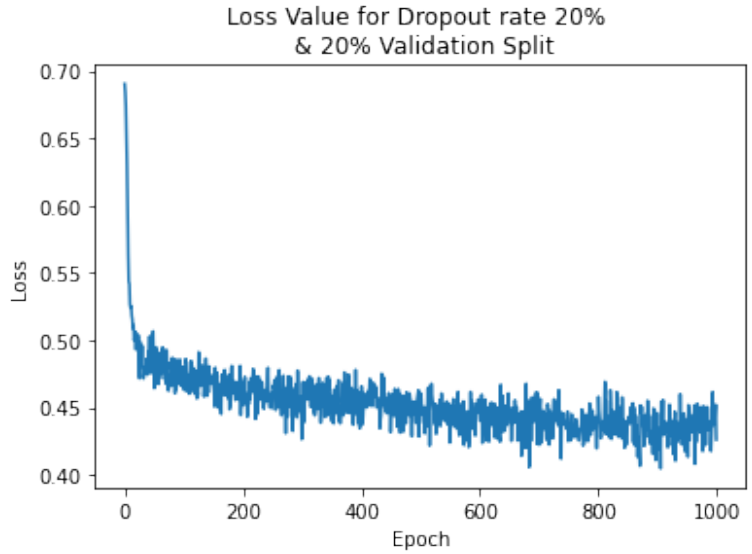
```
768/768 [==============================] - 0s 22us/step

accuracy: 81.90%
Confusion Matrix
================
True negatives:  455
False negatives:  94
False positives:  45
True positives:  174
```





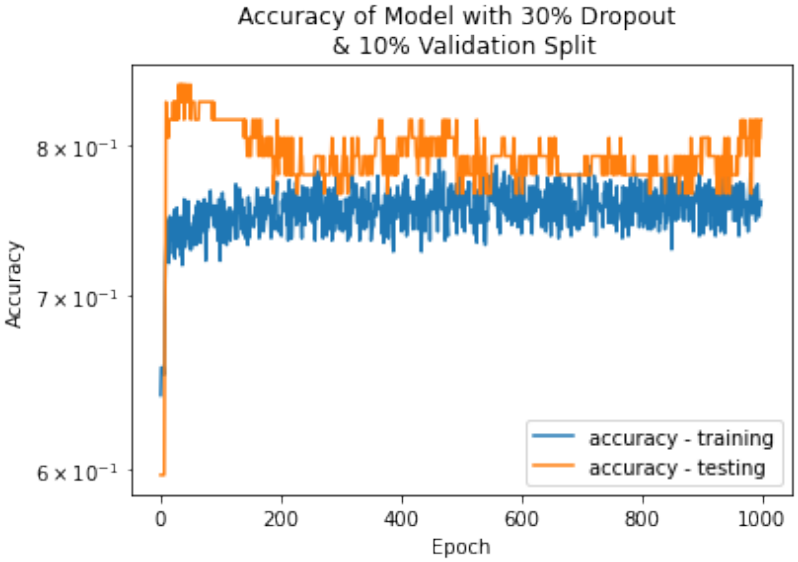
```
768/768 [==============================] - 0s 25us/step

accuracy: 80.86%
Confusion Matrix
================
True negatives:  446
False negatives:  93
False positives:  54
True positives:  175
```

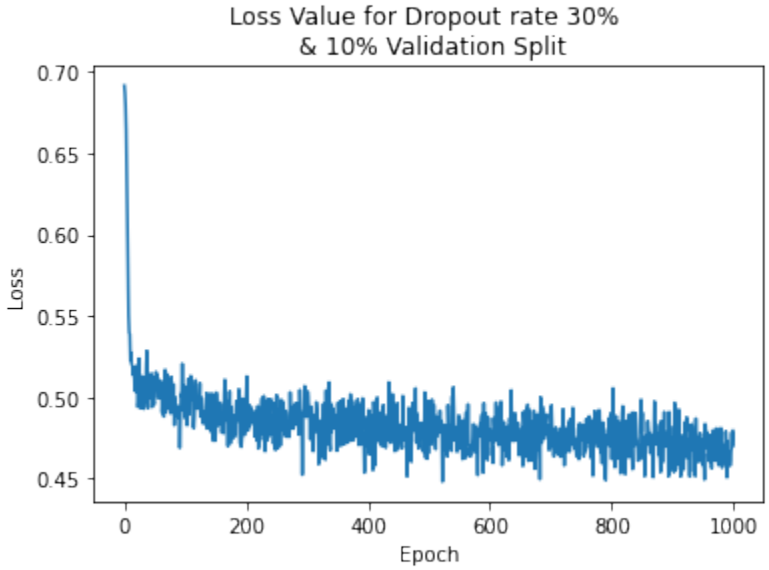




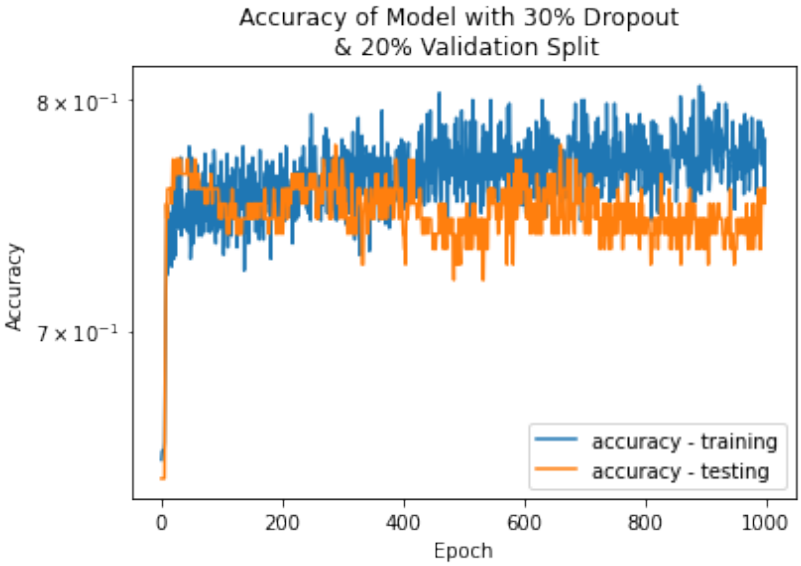
The final demonstration will be 30% dropout rate with 10%, 20%, 30% validation split.

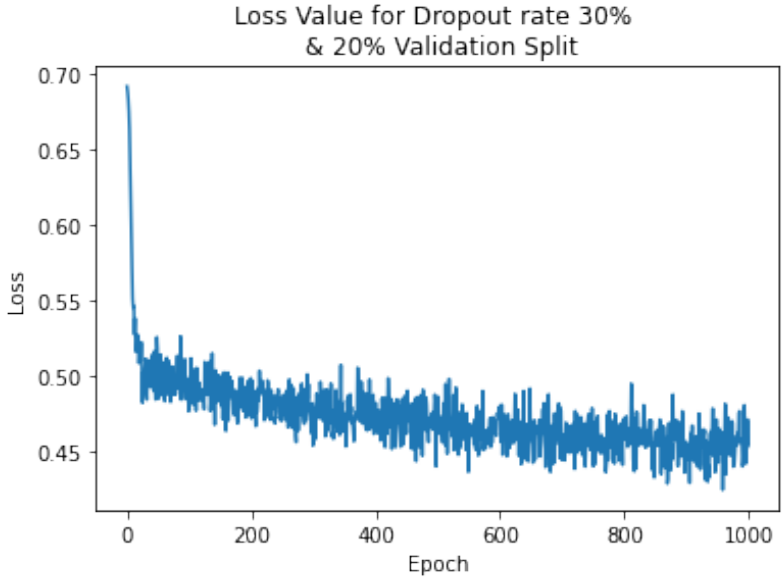
```
768/768 [==============================] - 0s 24us/step

accuracy: 81.51%
Confusion Matrix
================
True negatives:  455
False negatives:  97
False positives:  45
True positives:  171
```



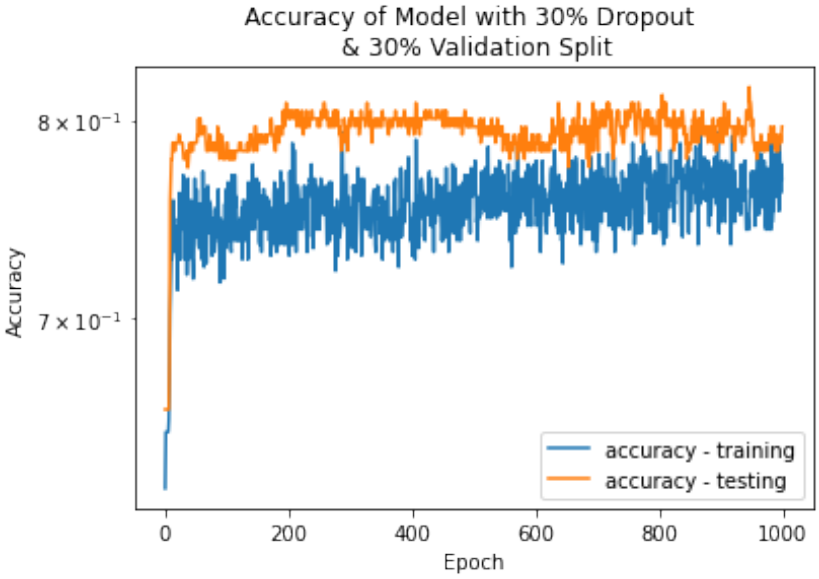


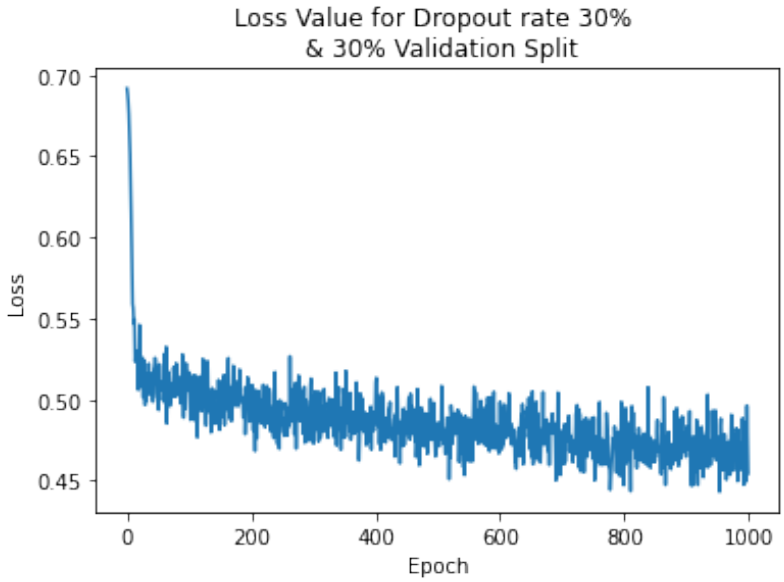
```
768/768 [==============================] - 0s 20us/step

accuracy: 81.38%
Confusion Matrix
================
True negatives:  463
False negatives:  106
False positives:  37
True positives:  162
```





```
768/768 [==============================] - 0s 23us/step

accuracy: 80.86%
Confusion Matrix
================
True negatives:  445
False negatives:  92
False positives:  55
True positives:  176
```





As the previous example without the checkpoint the network with the highest accuracy was the one with 10% dropout rate and 10% validation split. This could be for the same reason as the previous example. There isn't enough data to drop out many weights, the more training data

the better. There are many other way to adjust these networks to attain higher accuracy such as changing where the dropout is implemented or adding more dropouts. Adding more data points, standardizing the data differently, or having a backup test data set to truly test it and not use the original X data used in the example.