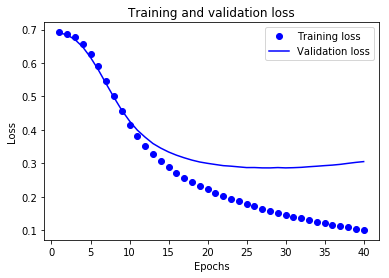
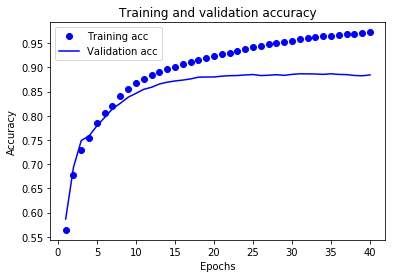
When we look at the graphs of the first file, two things are clear.

1. Training loss and Validation loss are similar for the initial few epochs and then training loss has become is at constant level whereas Validation loss has seen a steep decrease for each further epoch

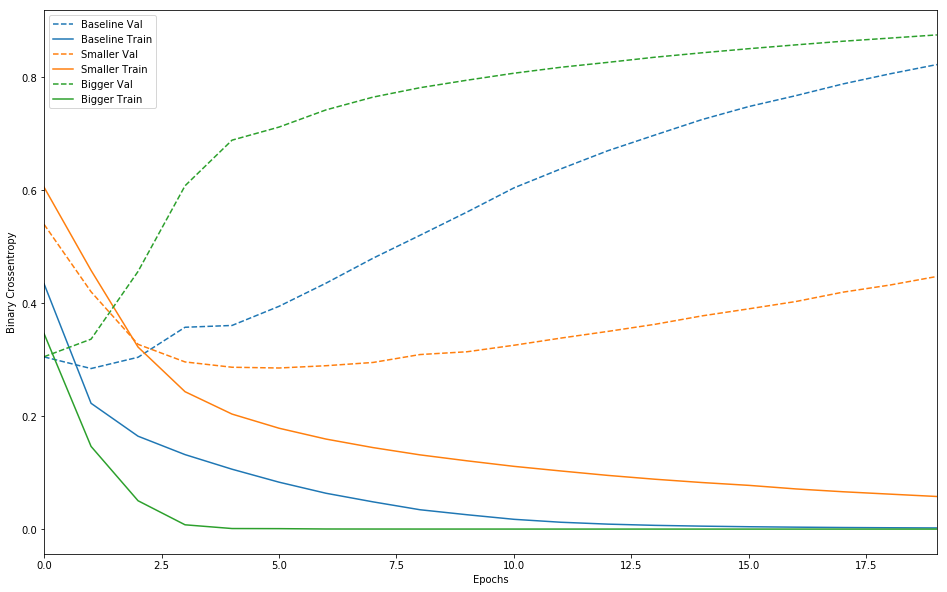


1. Similarly, Training accuracy and Validation accuracy are fairly similar for the first 10 epochs and then later on validation accuracy has become consistent at the same level and training accuracy has been increased for each further epoch. It means that validation is not decreasing or is constant then training accuracy increase for each further epoch



Similarly for graphs of second file, we can make the below observations

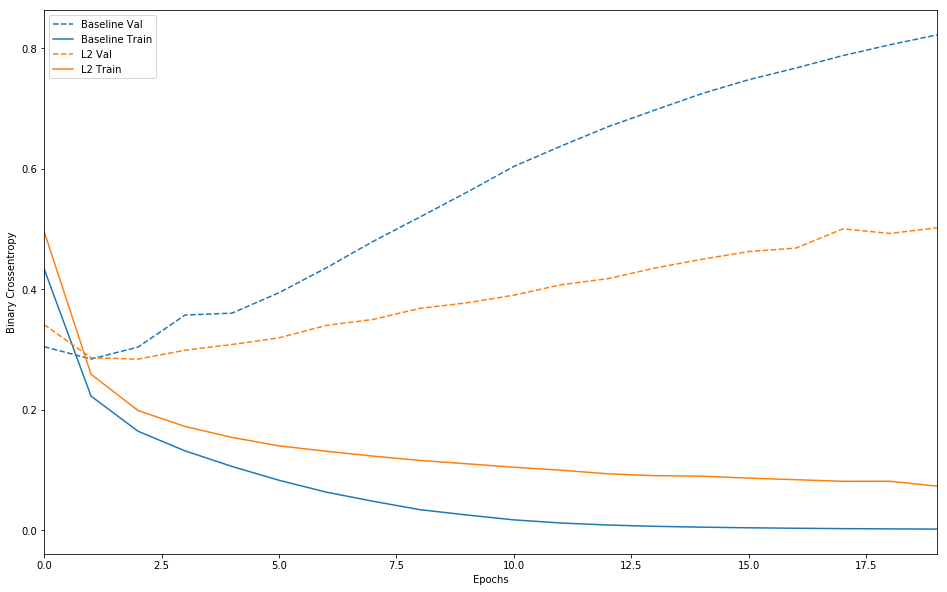
1. Except for the first few epochs, we can see that for each epoch Baseline Val increases whereas Baseline Train decreases
2. The trends are similar with Smaller and Bigger Val, Train values



**L2 Regularization**

1. Baseline Val is highest when Baseline Train is the lowest. The decrease in Val causes an increase in Train. Validation epoch went to peak after 10 epoch
2. Similarly L2 Val are Train are also similar for each further epoch .

This is a case of overfitting. The model should perform better on training data than the it has never seen earlier. Because when the model over optimizes, test data is not generalized. So we should stop training after 10 epochs.



**Dropout**

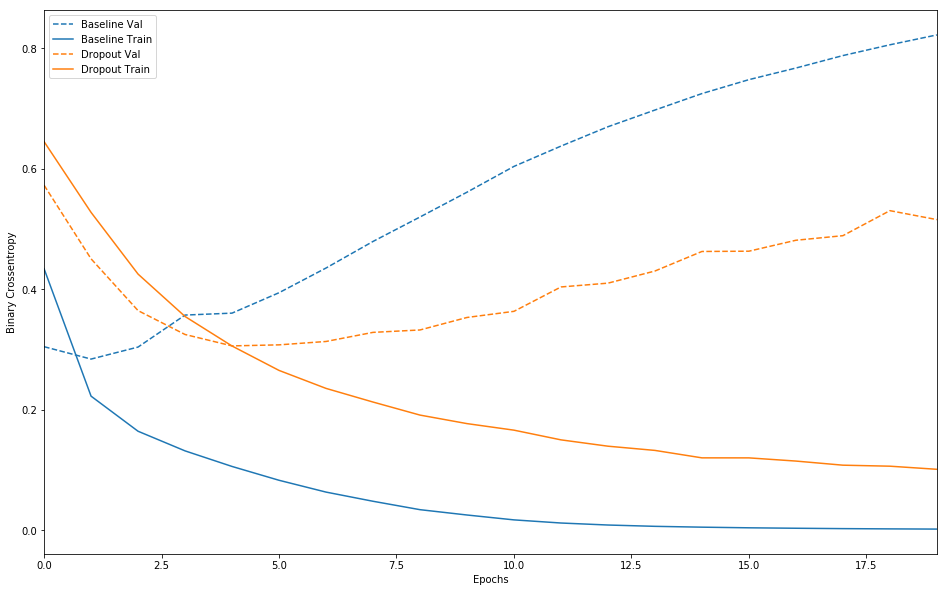
Gap between the training and validation defines the amount of overfitting.

In most cases dropout makes the training accuracy lower than it should be.

Also, if it is run for only 1-3 epochs, there can be strong overfitting as it is in first 1-2 epochs.

Overfitting also can appear when more parameters are used here to avoid underfitting. When we use larger dropout rates in the first few layers(hidden) it will alter the accuracy.

Also, since dropout ignores randomly selected nodes, if we run it again it might give a different accuracy with a great difference.



**Hidden Layer:**

Hidden layers are generally used to exhibit non-linear behavior. Ideally 2 hidden layers are used in any network. But the optimal number should be less the number of inputs. But if the training examples are plenty we should consider using more than 2 hidden layers. But we can use cross-validation to verify the optimal number of hidden layers.

Here we have used 16 hidden layers is justifiable because the batch size is big(512). The epoch count is also decided based on the regularization factor. But if the hidden layer size is decreased more than the optimal causes the decrease in accuracy. The validation is increased over the further epochs