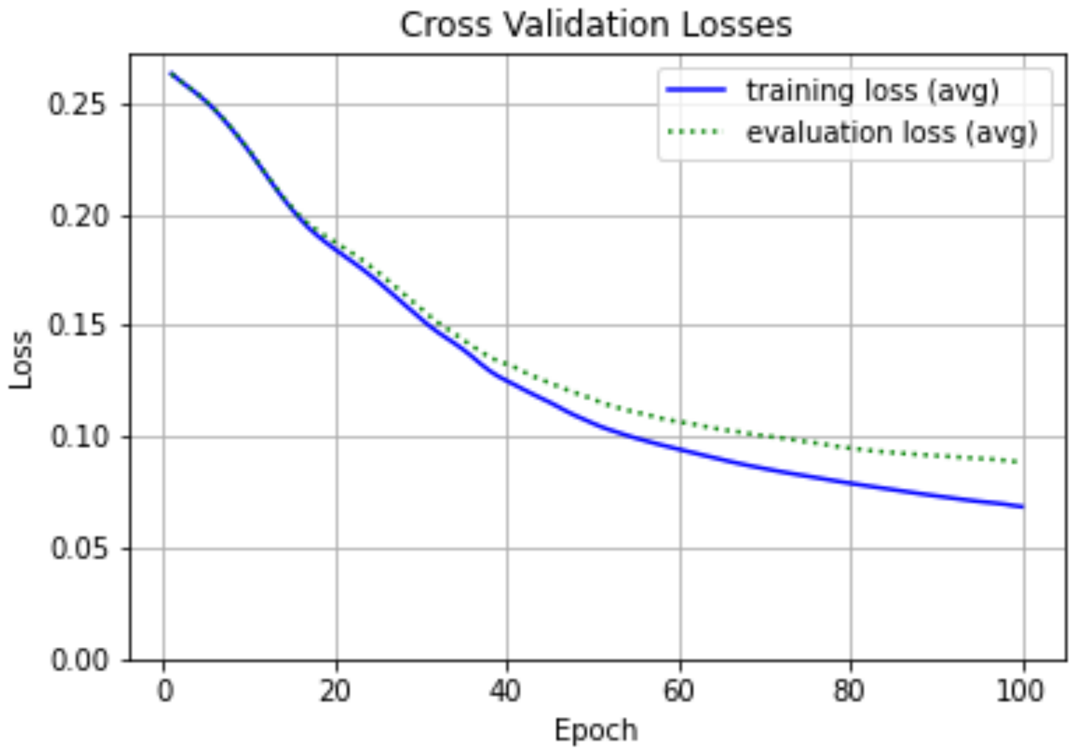
Strategy is to run 2-fold cross validation for 100 epochs and compute average losses to see how well a model generalizes. Once generalization is good enough, then I run train vs validation evaluation to see how well the model predicts.

Step 1: Cross validation loss on basic model

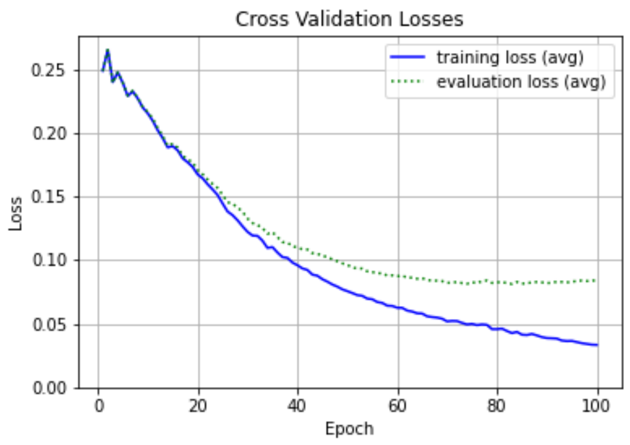


In this run, I had a simple model with the following structure:

1. 2D Average pooling layer that reduces the 24 x 24 image to a 12 x 12 image.
2. A fully connected layer with 7 nodes and ReLU activation
3. A second fully connected layer with 2 nodes and ReLU activation
4. A output layer with sigmoid activation.

My interpretation of this graph is that there is generalization error after about 40 errors and that the model overfits the data. The next step is to add a bit more power to see the if I could further reduce the loss on the cross-validation evalution set.

Step 2: Cross validation loss on model with 1 convulation layers

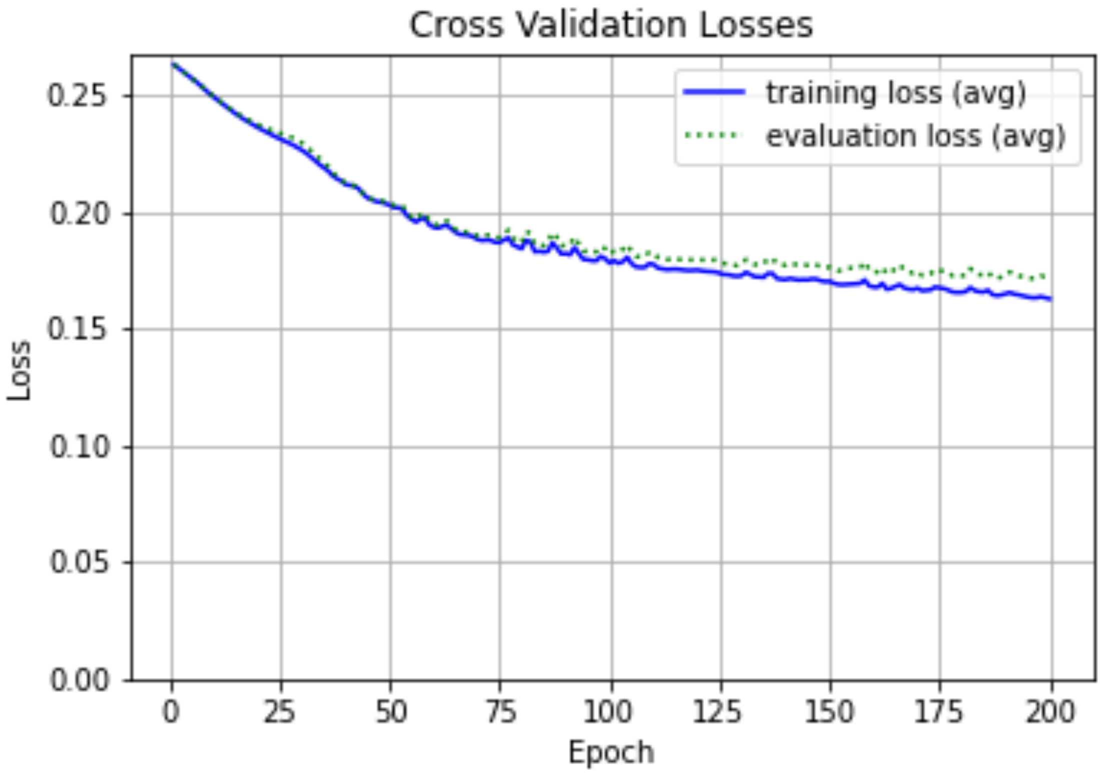


In this model, I add a single convulation layer and a single dropout layer to modify the structure as follows:

1. 1 Convulation layer with 5 nodes and kernel size of 3x3 and stride of 3x3.
2. 1 dropout layer with dropout probability = 50%
3. A fully connected layer with 9 nodes and ReLU activation
4. A second fully connected layer with 2 nodes and ReLU activation
5. A output layer with sigmoid activation.

In this run I see that although the loss on the evaluation set has decreased, the mode still overfits and is high bias. So the next step is to increase the power of the model a bit more. I have not shown all the results here, but I experimented with adding 2 max pooling layers and 2 dropout layers in the model. Finally I was able to get a model that generalizes well in Step 3.

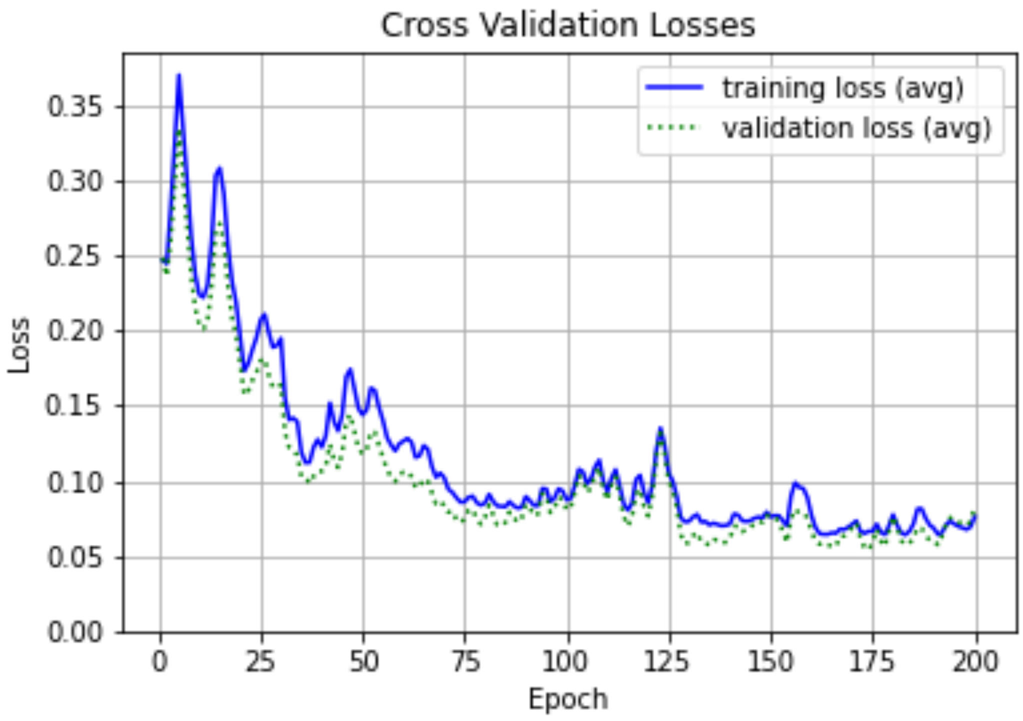
Step 3: Cross validation loss on model with 2 convulation layers



After some more experimentation, I was able to find a model that generalizes very well and does not overfit. I realize that because this is cross validation, the ration of data available to train and evaluate is 1:1. So I switch over to using train (90% of all data) vs validation set (5% of all data) for future experiments.
In this run, the model structure I got was:

1. A Convulation layer with 8 nodes and kernel size of 3x3 and stride of 3x3.
2. A Convulation layer with 4 nodes and kernel size of 3x3 and stride of 3x3.
3. A Dropout Layer with dropout probability = 50%
4. A fully connected layer with 9 nodes and ReLU activation
5. A second fully connected layer with 2 nodes and ReLU activation
6. A output layer with sigmoid activation.

Step 4 : Training vs validation Run with 2 convulation layers.



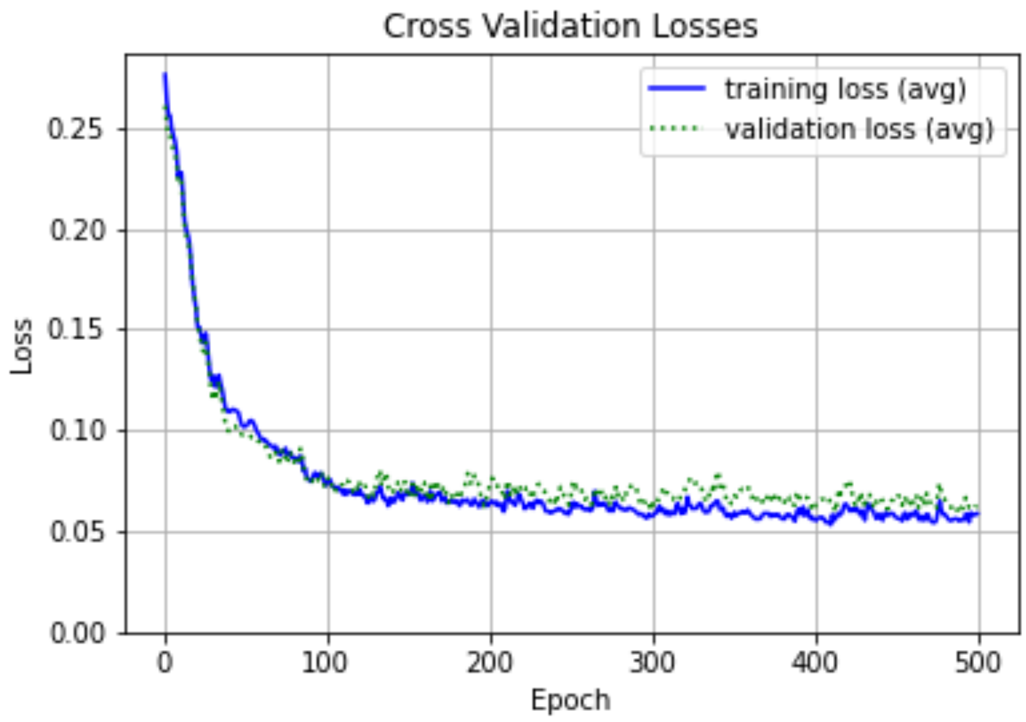
After I switch over to using training and validation sets for evaluation, I immediately see a high variance, although the model still generalized well and does not overfit. My interpretation is now to reduce the noise in the structure. I do that by adding a batch normalization layer before the convulation layers.

At this point the structure is as follows:

1. A Convulation layer with 8 nodes and kernel size of 3x3 and stride of 3x3.
2. A Convulation layer with 4 nodes and kernel size of 3x3 and stride of 3x3.
3. A Batch Normalization Layer
4. A Dropout Layer with dropout probability = 50%
5. A fully connected layer with 9 nodes and ReLU activation
6. A second fully connected layer with 2 nodes and ReLU activation
7. A output layer with sigmoid activation.

Since the output is very high variance, I experiment with positioning the batch normalization layer at a different position – beginning of the model.

Step 5 : Step 4 : Training vs validation Run with 2 convulation layers, 2 dropout layers and 1 batch normalization

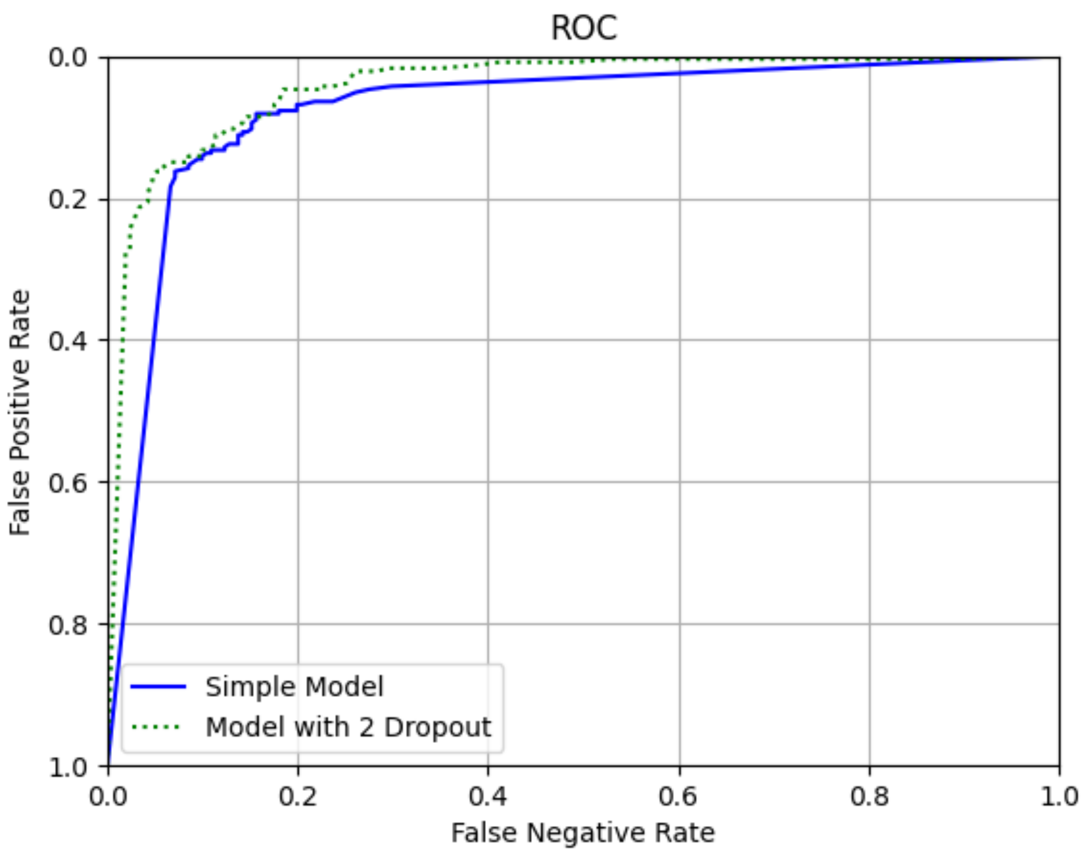


layer

Finally, I get a model that generalizes well and is not biased. The accuracies look in error bounds of each other after about 125 epochs, but there is still a trend towards decreasing loss even after 500 epochs. I believe if I let this model run longer, I will get better results. At this point, the model structure is as follows:

1. A Batch Normalization Layer
2. A Convulation layer with 8 nodes and kernel size of 3x3 and stride of 3x3.
3. A Convulation layer with 4 nodes and kernel size of 3x3 and stride of 3x3.
4. A Dropout Layer with dropout probability = 50%
5. A fully connected layer with 9 nodes and ReLU activation
6. A second Dropout Layer with dropout probability = 50%
7. A second fully connected layer with 2 nodes and ReLU activation
8. A output layer with sigmoid activation.

Step 6: ROC Plots beginning vs best on Test dataset



This ROC Plot shows the improvement I achieved with a convolution neural network over a fully connected neural network.