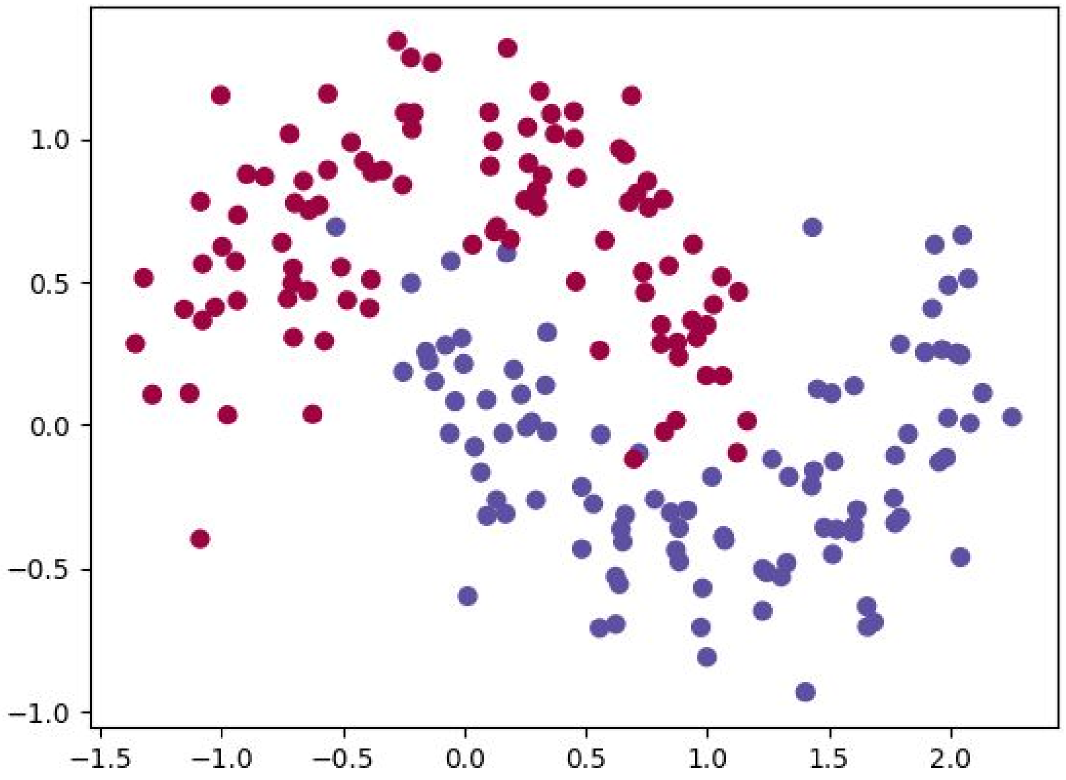
Abhijeet Mulgund (aam14)

COMP 576: Assignment 1

1. Backpropagation in a Simple Neural Network

a. Dataset



b. Activation Function

i. $\frac{d}{dx}tanh(x) = \frac{d}{dx}(\frac{sinh(x)}{cosh(x)}) = \frac{cosh(x)cosh(x) - sinh(x)sinh(x)}{cosh^2(x)}$

$= 1 - \frac{sinh^2(x)}{cosh^2(x)} = 1 - tanh^2(x)$

ii. $\frac{d}{dx}sigma(x) = \frac{d}{dx}(1+e^{-x})^{-1} =- (1+e^{-x})^{-2}\frac{d}{dx}(1+e^{-x})$

$= -(1+e^{-x})^{-2}(-e^{-x}) = (\frac{1}{1+e^{-x}})(\frac{e^{-x}}{1+e^{-x}}) = (\frac{1}{1+e^{-x}})(\frac{1+e^{-x}-1}{1+e^{-x}})$

$= (\frac{1}{1+e^{-x}})(\frac{1+e^{-x}-1}{1+e^{-x}}) = (\frac{1}{1+e^{-x}})(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}})$

$= sigma(x)(1 - sigma(x))$

iii. $\frac{d}{dx}ReLU(x) = \frac{d}{dx}max(0,\ x) = \frac{d}{dx}0$ if $x \le 0$ else $\frac{d}{dx}x$

$= 0$ if $x \le 0$ else $1 = max(0,\ sign(x))$

c. **Build the Network** (Code)

d. **Backward Pass - Backpropagation**

(Assuming n sample input)

i. $\frac{dL}{dz_2} = \delta_3 = \hat{p} - y$

ii. $\frac{dL}{dW_2} = \frac{dz_2}{dW_2}\frac{dL}{dz_2} = \frac{d}{dW_2}(a_1 W_2 + b_2)\,\delta_3 = a_1{}^T \delta_3$

iii. $\frac{dL}{db_2} = \frac{dz_2}{db_2}\frac{dL}{dz_2} = \frac{d}{db_2}(a_1 W_2 + b_2)\,\delta_3 = (1)_{1xn}\delta_3$

iv. $\frac{da_1}{dz_1} = da_1$

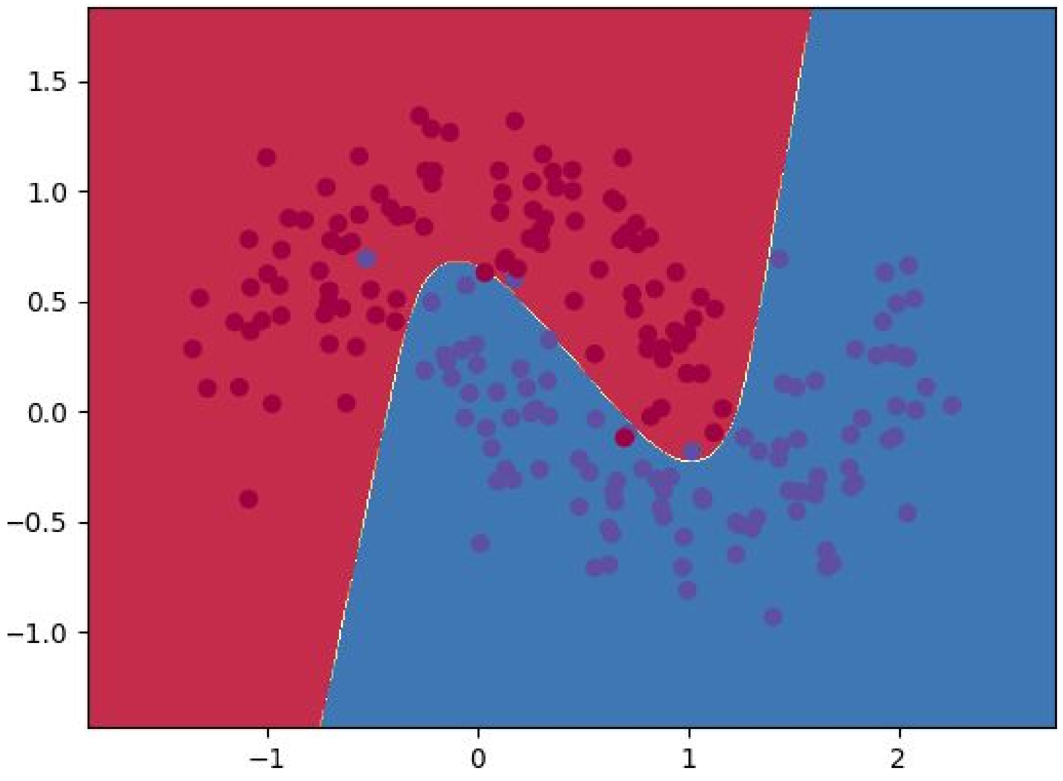
v. $\frac{dL}{dz_1} = \delta_2 = \frac{dL}{dz_2}\frac{dz_2}{da_1}\frac{da_1}{dz_1} = \delta_3 \frac{d}{da_!}(a_1 W_2 + b_2) * da_1 = \delta_3 W_2{}^T * da_1$

vi. $\frac{dL}{dW_1} = \frac{dz_1}{dW_1}\frac{dL}{dz_1} = \frac{d}{dW_1}(XW_1 + b_1)\,\delta_2 = X^T \delta_2$

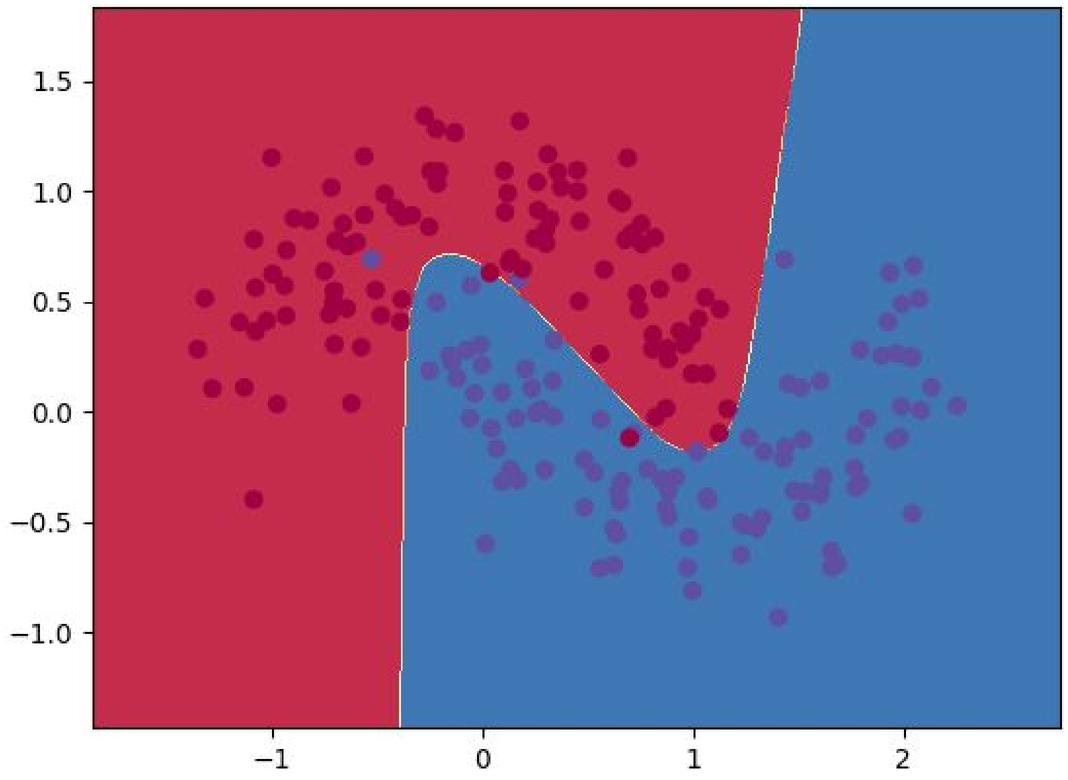
vii. $\frac{dL}{db_1} = \frac{dz_1}{db_1}\frac{dL}{dz_1} = \frac{d}{db_1}(XW_1 + b_1)\,\delta_2 = (1)_{1xn}\delta_2$

e. **Time to Have Fun - Training!**

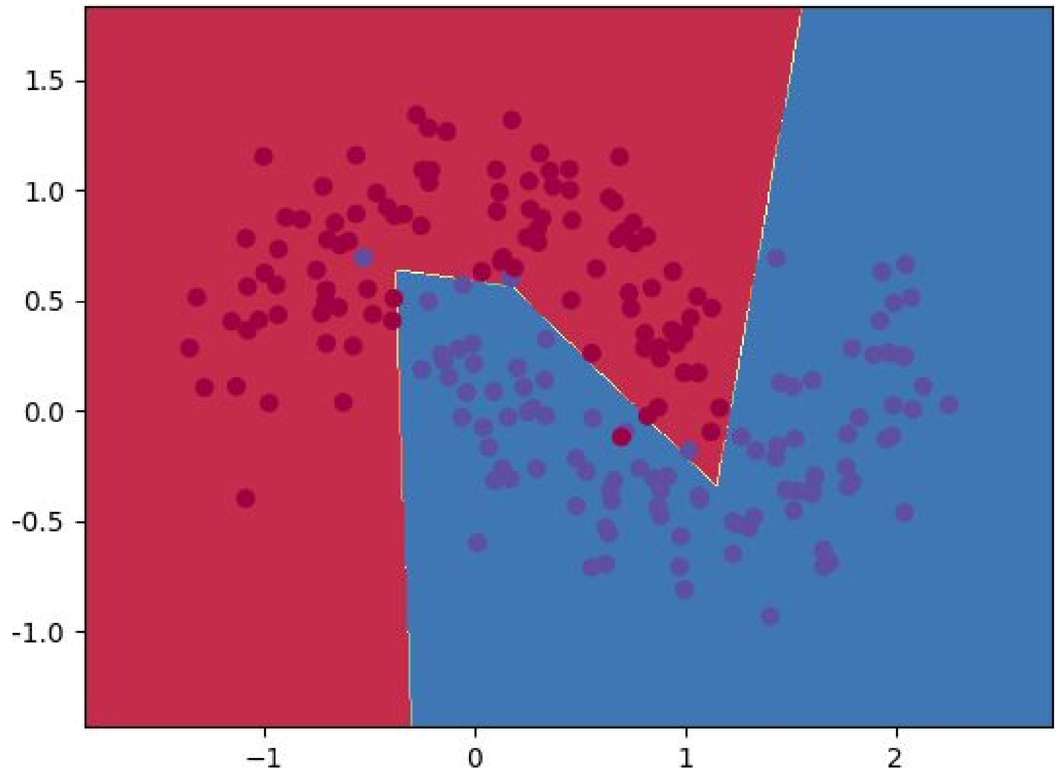
i. Tanh - 3 hidden



ii. Sigmoid - 3 hidden

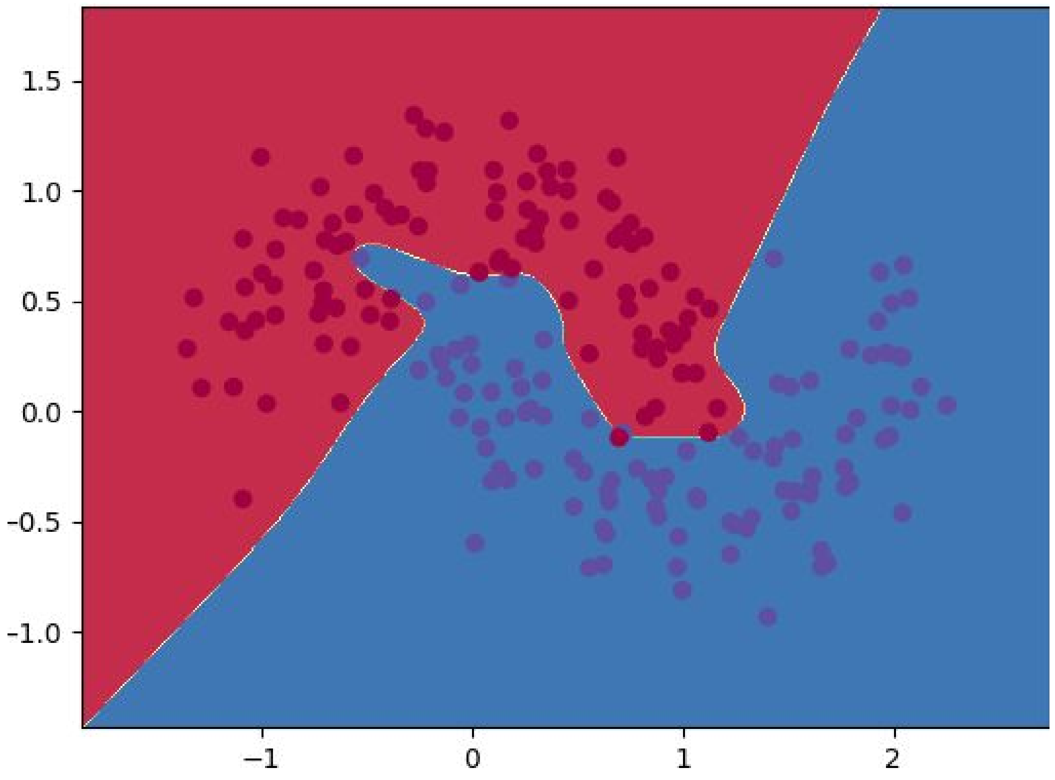


iii. ReLU - 3 hidden



The *tanh* and *sigmoid* have very similar decision boundaries; however, the *tanh* finished at 0.7076 loss and the *sigmoid* finished at 0.07879 loss. The *relu* had a very jagged decision boundary. This makes sense, since the *relu* activation is just a piecewise linear function. It finished with 0.07122 loss.

iv. Tanh - 100 hidden



With 100 hidden neurons, the neural net seems to try to grab some outliers in the data and overfits compared to when it had 3 hidden neurons. The loss was much lower at 0.03249.

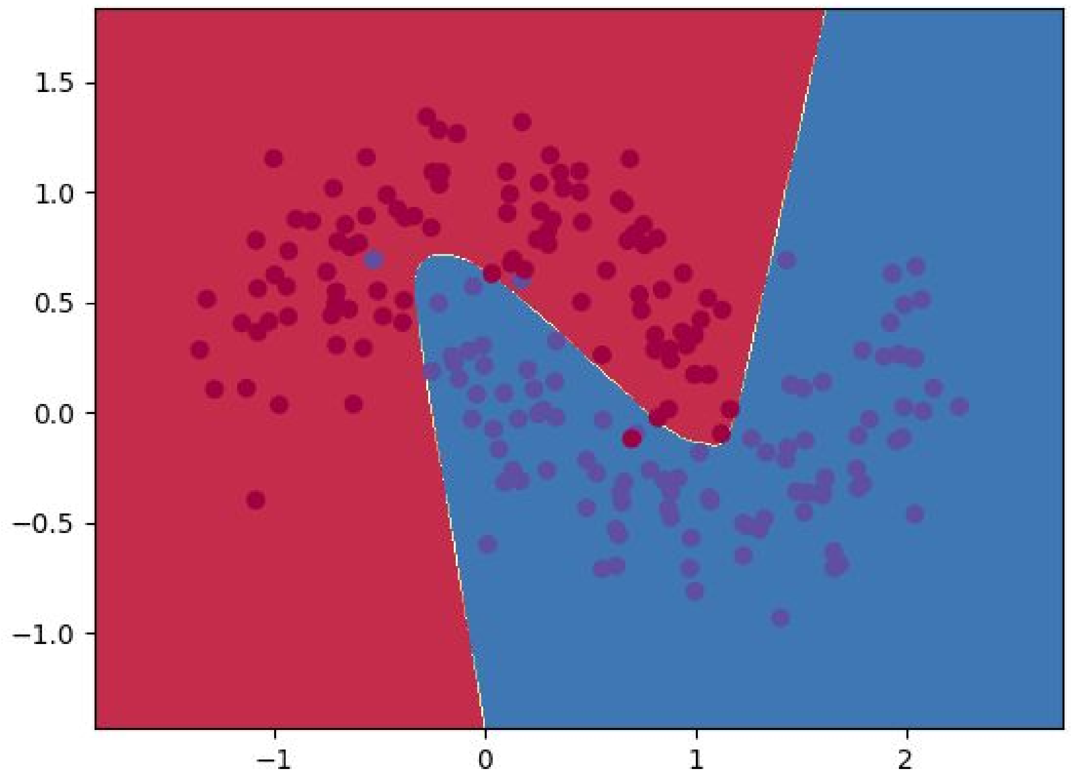
2. **Even More Fun - Training a Deeper Network!!!**
    a. Design: As constituents of the neural network, I made a DenseLayer class (simple fully connected layer with an activation) and a SoftmaxLayer class (fully connected layer

that computes the softmax on the output). These layers had the following parameters

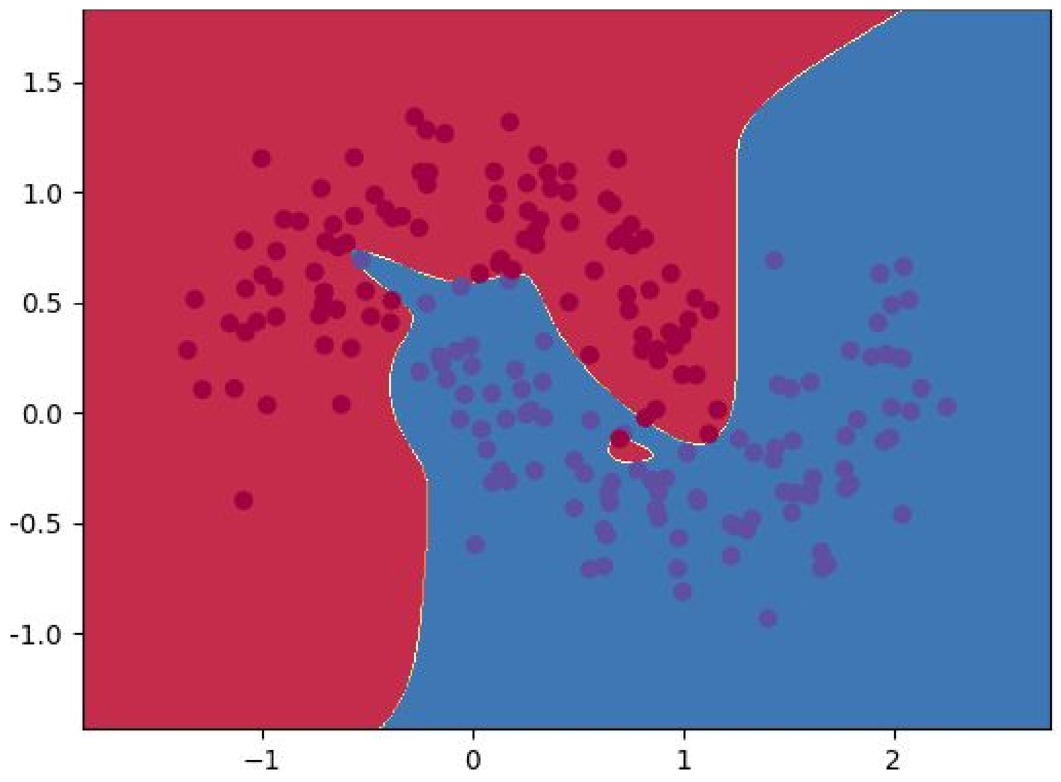
i. Input Size
ii. Output Size
iii. Activation (only DenseLayer)
iv. Regularization

This way, all properties of the neural network are defined on a per layer basis. All the NeuralNetwork class did is sequentially run a bunch of layers. The layers performed their own forward pass (returning their output) and backward pass (returning their delta).

b. Tanh - hidden 3-3-3

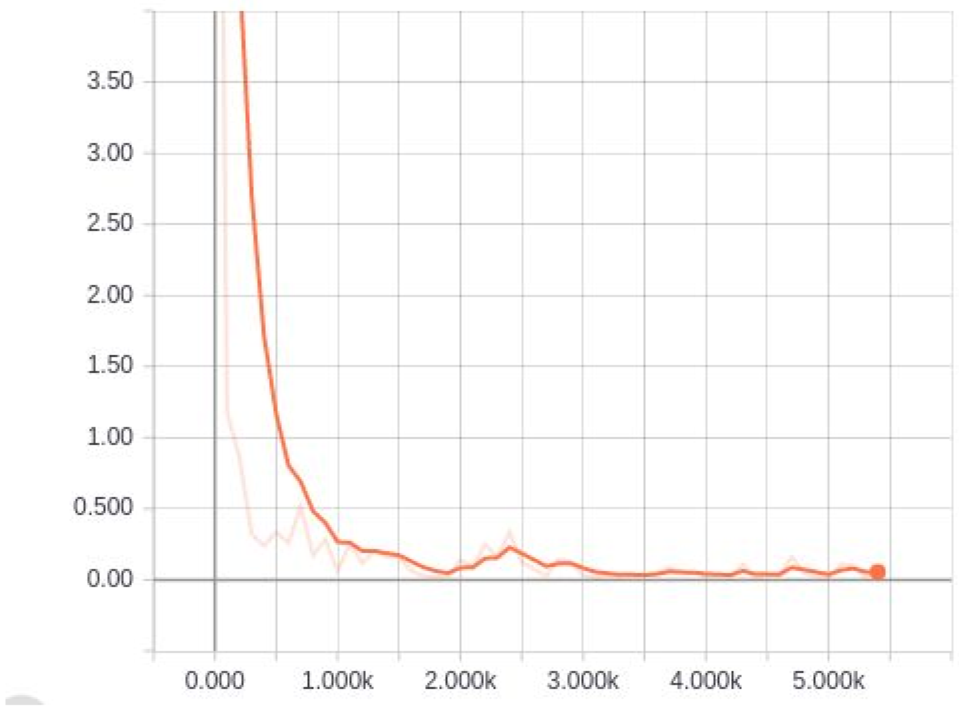


c. Sigmoid - hidden 10-6-4-3-3

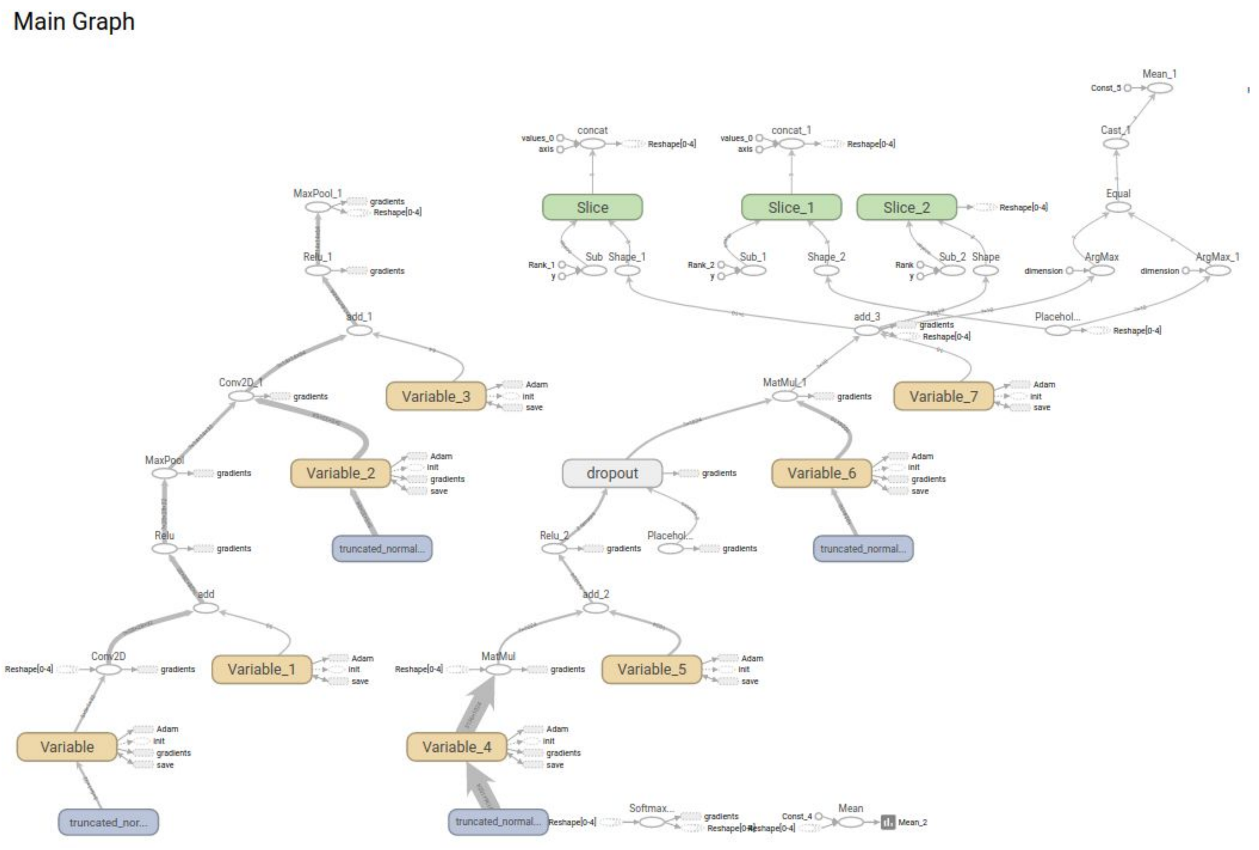


Just changing to 3 layers with tanh doesn't make too much of a difference in the decision boundary. However, as I increase the number of layers and neurons, you can see with the second diagram that rampant overfitting starts. So much so that it even tries to capture each outlier point with a small island.

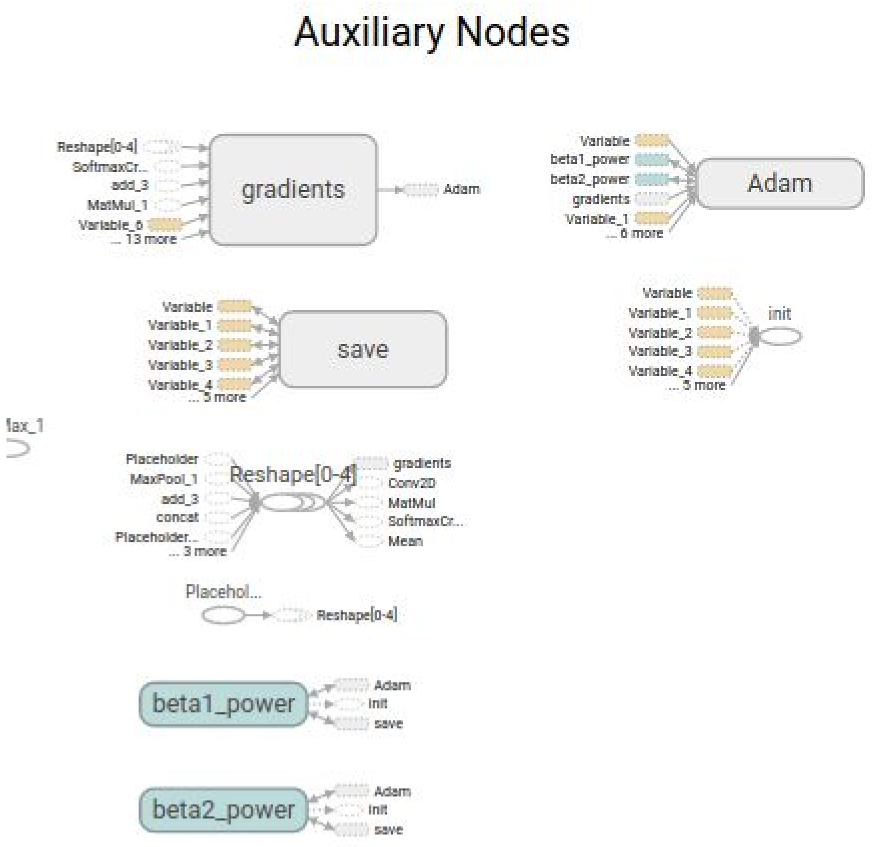
3. **Training a Simple Deep Convolutional Network on MNIST**
    a. **Build and Train a 4-layer DCN**
        i. Loss wrt epoch



ii. Main Computational Graph

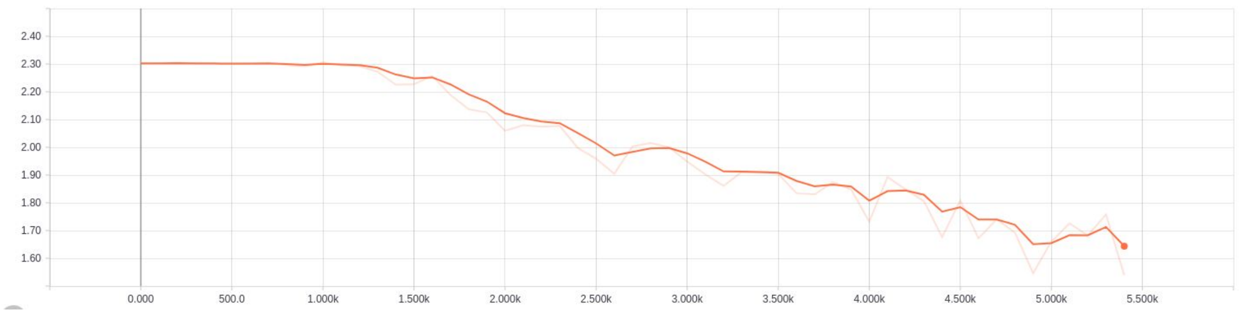


iii. Auxilary Nodes



b. Changed all the activations to softmaxes

i. Loss wrt to epoch



ii. T-SNE of weights of fully connected layer