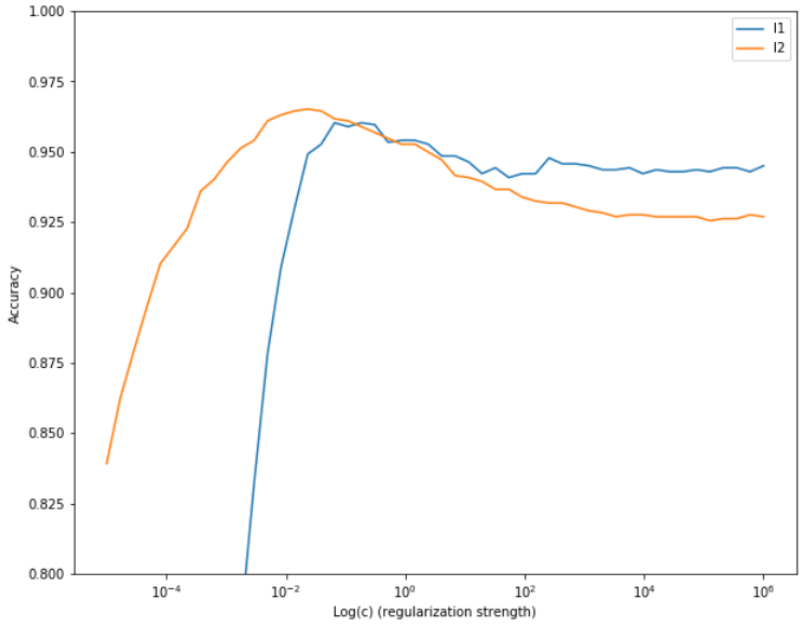
Part 1

Accuracy:



Conclusion

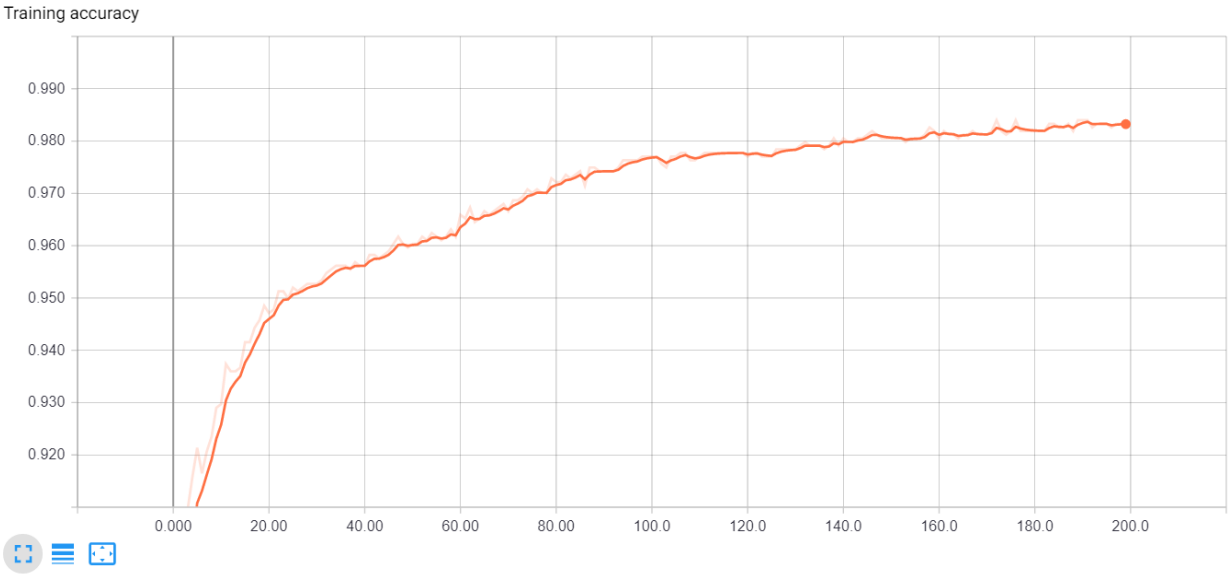
As we see in the plot above, for l1 regularization, the peaks appears at c = 0.06. For l2 regularization, the peak appears at about c = 0.02. They are near 10^{-2}

Therefore, I would choose c = 10^{-2} as the regularization strength

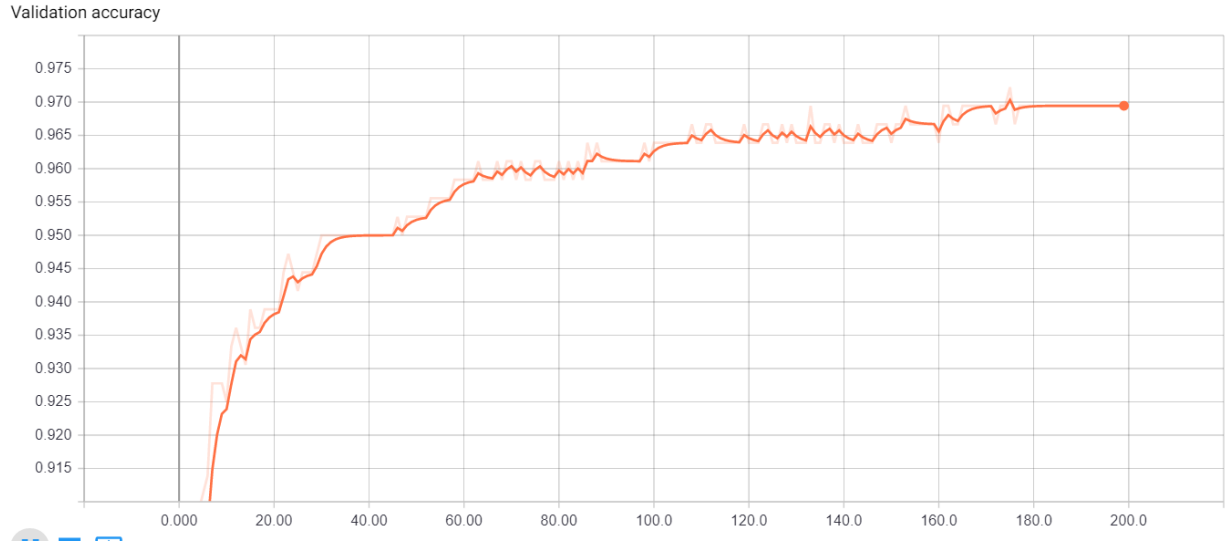
Part 2

2.1 Softmax regression

Training accuracy

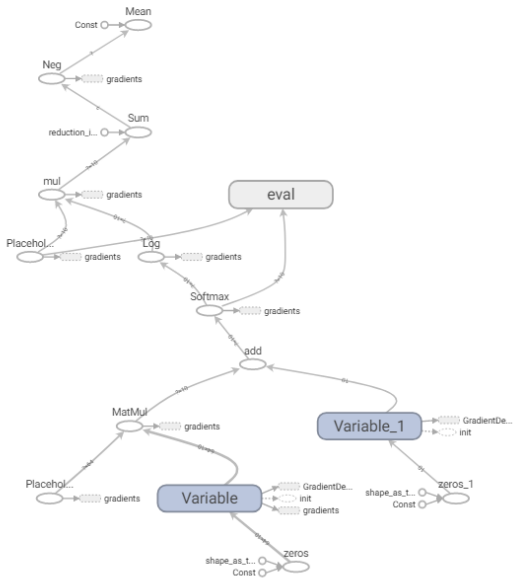


Validation accuracy



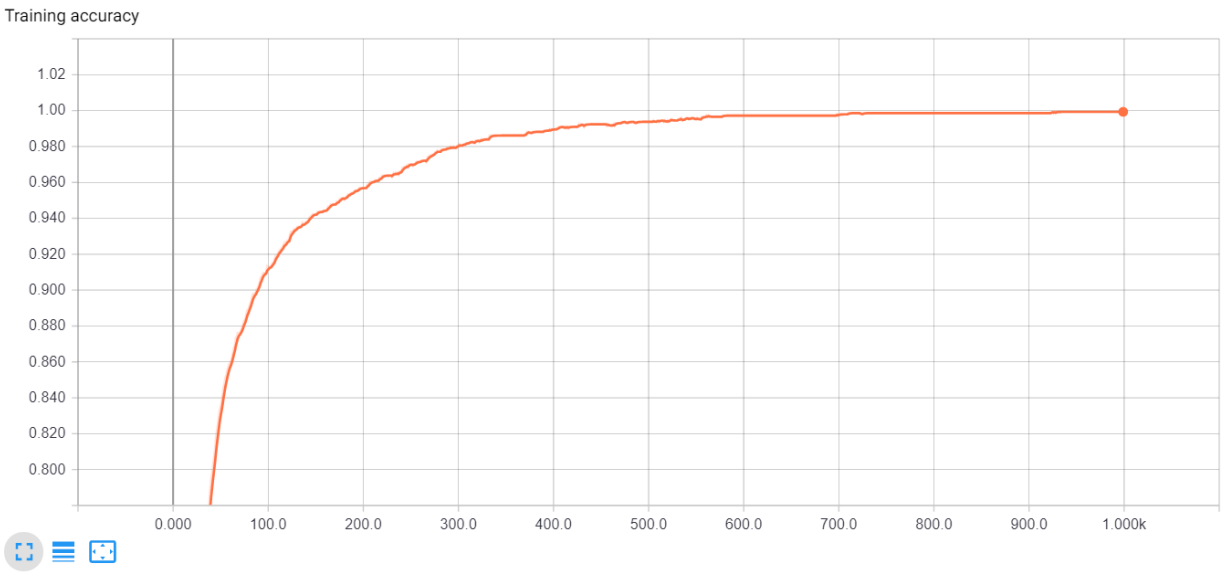
As we see, the training accuracy reaches 0.983 and the validation accuracy reaches 0.97

Neural Net Graph

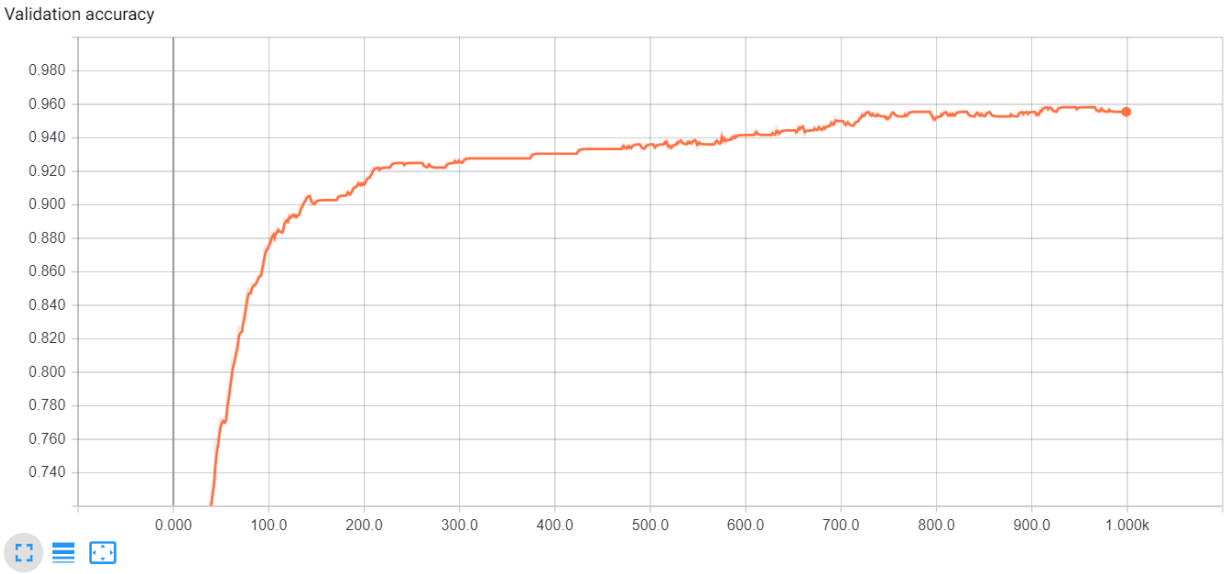


2.2 Dense Neural Net

Training accuracy

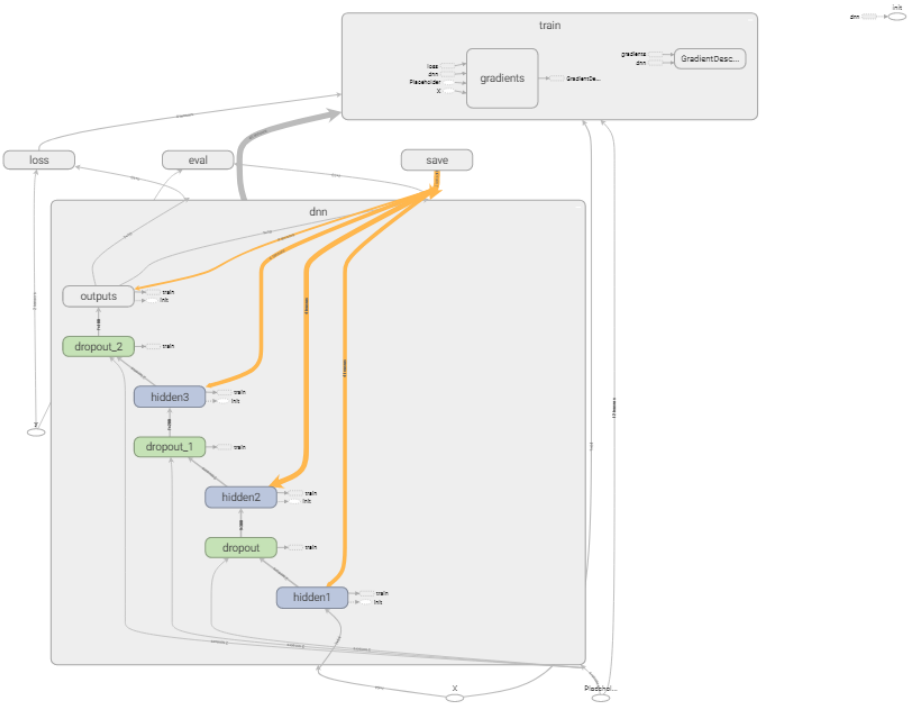


Validation accuracy



As we see, since we have 3 layers and the number of epoch are 1000, the training accuracy reaches 1, the validation accuracy reaches 0.96

Neural Net Graph



Comparison

The Dense Neural Net is much more complicated than the multiclass logistic regression.

The complexity for Dense Neural Net:

weights = 64*300* + *300*200 + 200*100* + *100*10 = 100200

biases = 300 + 200 + 100 + 10 = 710

total = 100910

The complexity for the multiclass logistic regression:

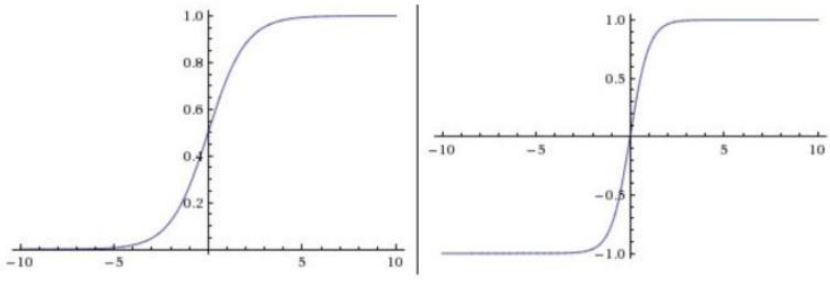
weights = 64*10 = 640

biases = 10

total = 650

Part 3

3.1



Neuron Saturation

Neuron saturation means activation functions take on values that are close to the boundaries of this range.

If we use the activation function like Sigmoid or Tanh, when the input value is close to the boundaries, say, 0 or 1(as we see in the figure above), the gradient $\frac{\partial y_j}{\partial x_j}$ would diminish (near 0) when doing the back propogation. Therefore, if the gradient is very small, the multiplication would be near 0. The moment that happens, the learning process stops and the weights stop updating with the iterations.

If the network is absolutely correct on all inputs and does not need to change, then saturation would be ok. But if the neurons were wrong, it becomes a big problem. This is especially relevant at initialization.

Batch normalisation

Batch normalization is a technique for imporving the performance and stability of neural networks, and also makes more sophisticated deep learning architectures work in practice.

The way to do this is to normalise the inputs of each layer so they have a mean output activation of zero and stadnard deviation of one. The method is shown as below:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \quad mini - batch - mean$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \quad mini - batch - variace$$

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad normalize$$

$$y_i = \gamma\sigma_B^2 + \beta = BN_{\gamma,\beta}(x_i) \quad sacle - and - hift$$

How does the batch normalization help:

The nature of neural net is to study the data distribution, once the distribution of the training data is different from the test data, then the generality would decreased a lot. Besides, once the distribution of each training batch set is differnt, then the network has to learn it in each iteration, this would greatly slow down the training speed, that's why we need to do the batch normalization.

The neural net training is a very complicated process. Minor change in the first few layers can accumulate in the next few layers and magnify a lot. For example, the coefficents in the first layer keep changing during the process, the coefficients in the second layer would definitly change with the change of the first layer. So what the batch normalization does it to normalize the output of former layer then make them as the input of next layer.

The advantages of the batch normalization can be concluded as follows:

1. A large learning rate can be chosen, which would speed up the training process.
2. We don't needto deal with the drop out and L1 or L2 regularization. Or we can choose smaller regularization strength.
3. Make weights easier to initialize. Batch normalisation helps reduce the sensitivity to the initial starting weights.
4. Make more activation functions viable. For example, Sigmoid function loses gradient quickly, which means it can’t be used in deep networks, and ReLU often dies out during training. Batch normalization regulates the values going into each activation function, nonlinearities that don't work well in deep networks tend to become viable again.

3.2

Activation function

Activation functions are really important for a Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable.

It introduces non-linear properties to the network. The main purpose is to convert an input signal of a node in a A-NN to an output signal. That output signal now is used as a input in the next layer in the stack.

If we don't apply an activation function then the output signal would be a simple linear function. A linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data.

Now that we need a Neural Network to learn and represent almost anything and any arbitray complex funtion which amps inputs to outputs. Hence using a non-linear activation we are able to generate non-linear mappings from inputs to outputs

Sigmoid

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

The range of the function is between 0 and 1. It is a S-shaped curve as shown curve.

1. vanishing gradient problem
2. the output is not zero centered. It makes the gradient updates go too far in different directions
3. saturating and kill gradient
4. small convergent rate

Tanh

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

1. output centers at 0, ranging between -1 and 1.
2. Having stronger gradients
3. Avoding bias with those gradients

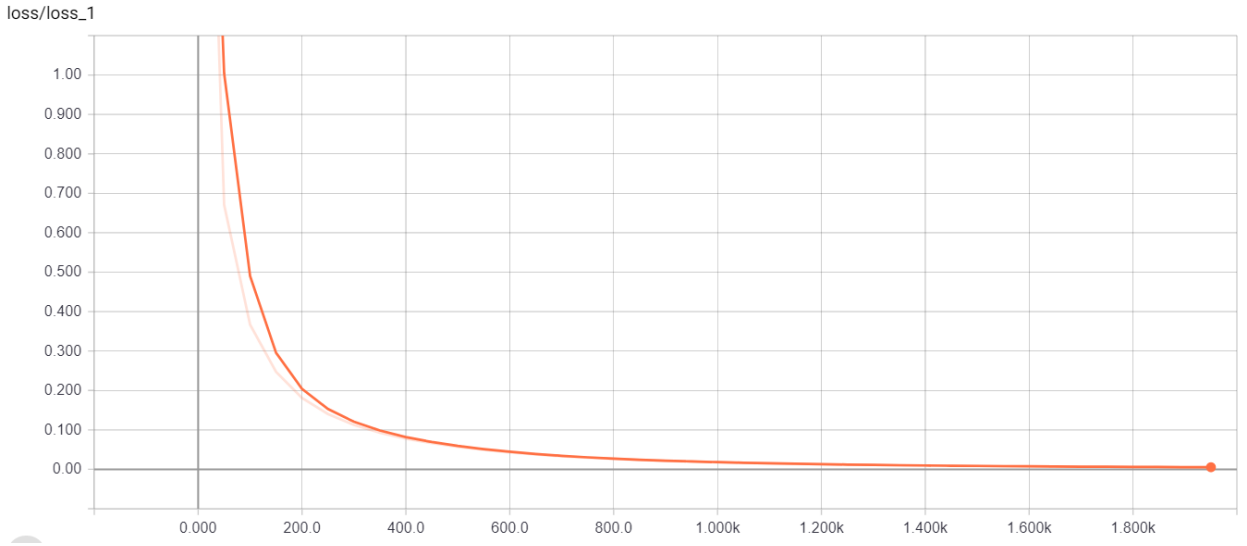
Relu

$$f(x) = max(0, x)$$

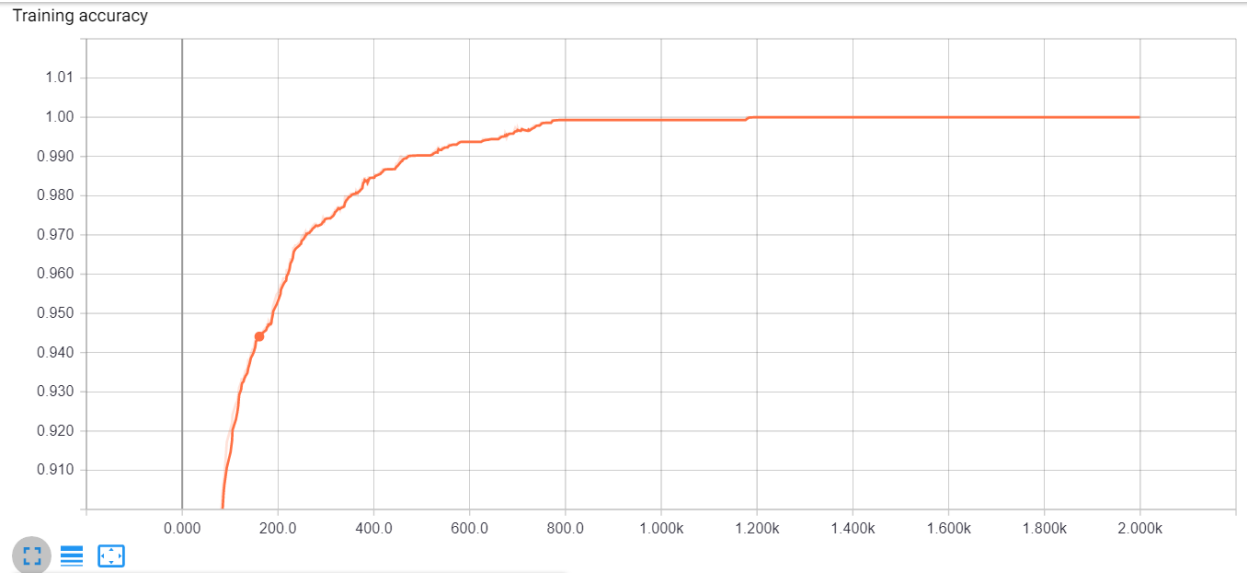
1. simple and efficient
2. reduced likelihood of the gradient to vanish
3. sparsity arises when x <= 0
4. it should only be used with hidden layer of the neural network model
5. somtimes ReLu could result in Dead Neurons.

3.3

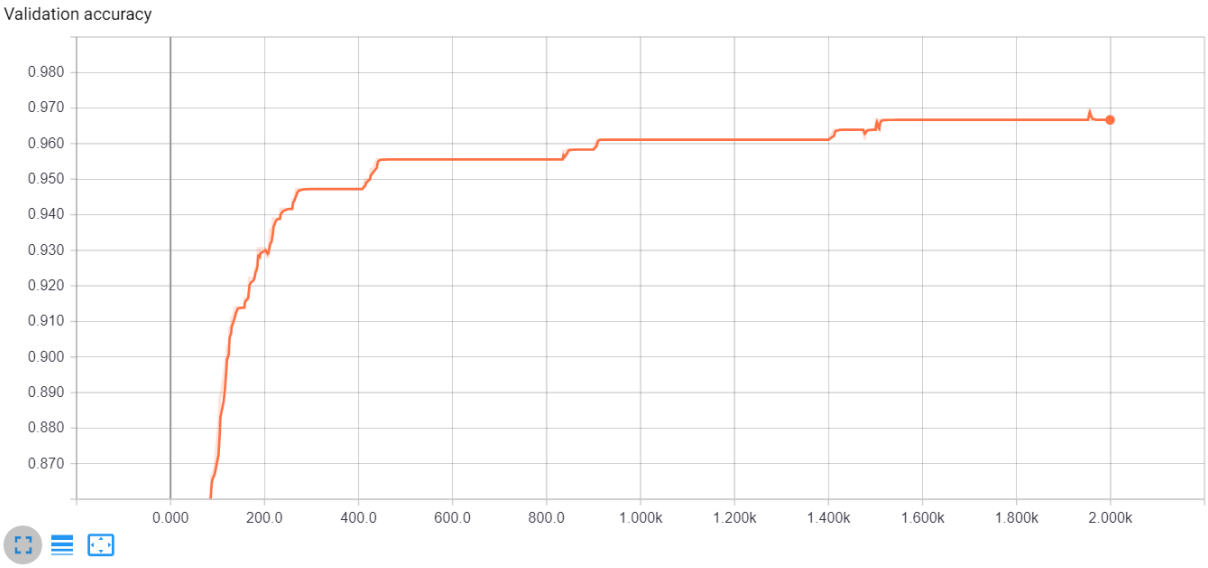
Training loss



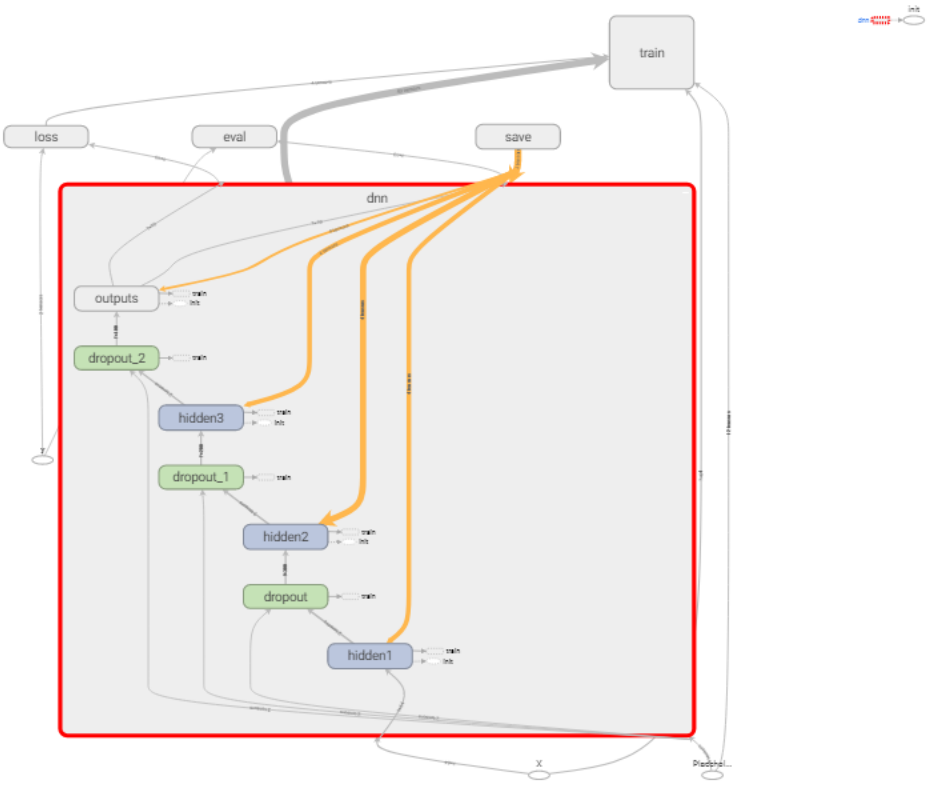
Training accuracy



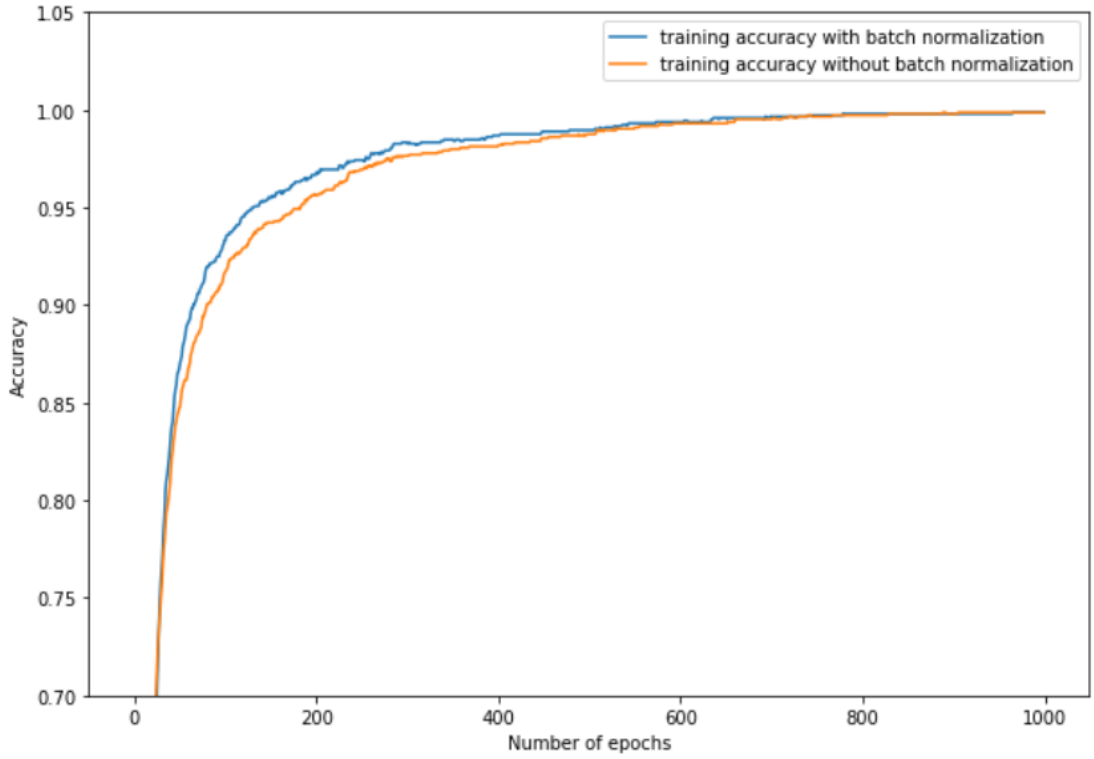
Validation accuracy



Neural Net Graph



Training accuracy: batch normalization vs without batch normalization



Conclusion

As we see from the plot above, the method with batch normalization achieve higher accuarcy in terms of both training and test accuracy. The training rate is also faster.