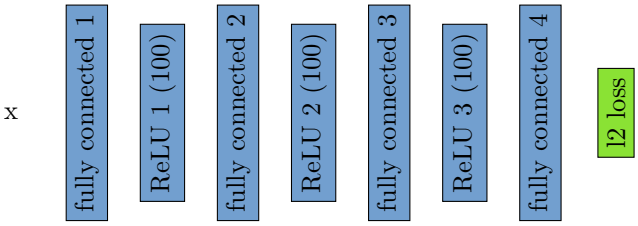
Exercise 3: Optimization

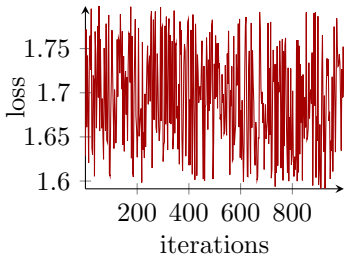
Name: ..... UTID: .....

You decided that solving these qizzes by hand simply does not cut it anymore. Inspired by the success of deep networks in computer vision you decide to train a deep network to answer simple quiz questions for your. The input is a screenshot x ($45 \times 45 \times 3$ image) of the particular question you want answered, the output is a scalar (positive for the "yes", negative for "no"). After creating and answering thousands of mock quizzes (as training data), you design a simple network to do all the heavy lifting for you.



However, during training you hit a few hiccups.

1. You train your network, but the loss does not decrease at all. You decide to plot the loss show below.



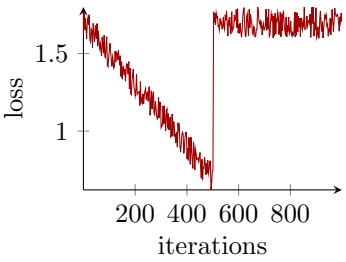
a) Why does the loss fluctuate between iterations?

b) Why might the loss not decrease? (check any that apply)

- ☐ The network is too large. Reduce the number of parameters.
- ☐ You picked the wrong non-linearity.
- ☐ You didn't initialize your network properly.
- ☐ Your learning rate is too low.
- ☐ Your learning rate is too high.
- ☐ You initialized the last layer to zero, which prevents gradients from through the network.
- ☐ You initialized the first layer to zero, which prevents gradients from through the network.

2. After you address all the above issues you move on. Now the network trains, however after a certain while the loss suddenly increases and the network stops training.

Upon further investigation you realize that the output of the network is constant after 500 iteration (but not before that). In fact, at iteration 500 most of the ReLU activations die in the network.

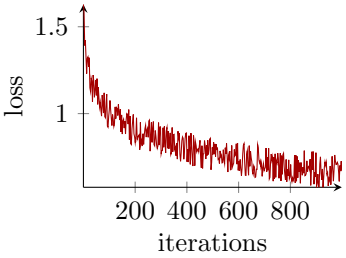


a) Briefly explain what a dead activation is and why it is undesirable?

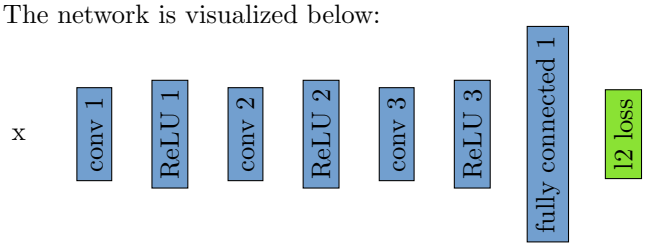
b) How can you fix the training? (check any that apply)

- ☐ You picked the wrong non-linearity.
- ☐ You didn't initialize your network properly.
- ☐ Your learning rate is too low.
- ☐ Your learning rate is too high.
- ☐ Nothing is wrong. Just stop training after 499 iterations.

3. After debugging your network for hours you are finally able to train it. However, the fully connected network does not perform very well. You suspect that the first two layers are unable to capture the complexity of the quiz questions in just 100 hidden units. In order to expand the size and representative power of the network you decide to switch to convolutions (5×5 kernels, stride 2, no padding). Each convolution outputs 100 channels.



The network is visualized below:



a) What is the output dimension of each activation map in the network, given a $45 \times 45 \times 3$ input?

b) How many parameters did the convnet use? How many parameters would an equivalent fully-connected network use? Ignore biases.

After you trained your convolutional network, it fits the training data perfectly. However, the performance on unseen question is not much better than chance. In short, the model overfits. Class today and Thursday will tell you how to prevent this.