What we need for for our neural network:

* 3 layers (an input, hidden and output layer)
* Input layer:
  + n neurons
* Hidden layer:
  + dh neurons
    - with a rectifier (RELU) as a non linearity
* Output layer:
  + m neurons
    - with a numerically stable softmax non linearity
      * Refer to lecture notes for a proper way of computing numerically stable softmax
* Parameters:
  + Between layer 1 and 2:
    - n x dh weights
    - dh biases (one per neuron in layer 2)
  + Between layer 2 and 3:
    - dh x m weights
    - m biases (one per neuron in layer 3)
  + Initialization:
    - Weights: [-1/(sqrt(nc); -1/(sqrt(nc)] where nc is the number of inputs for **this layer.**
    - Biases: 0
* fprop and bprop:
  + fprop will compute the forward propagation and the cost
  + bprop will use computed activations by fprop and does the vakpropagation of the gradients from the cost to the input following precisely the steps derived in part 2
* Finite difference gradient check:
  + Calculate the value of the loss function for the current parameter values
  + For each parameter, change the value by adding a ksi between 10-6 and 10-4 and calculate the new value of the loss (same example or mini-batch)
  + Set parameters back to what they were
  + divide the change in the loss function by ksi
  + The ration of your gradient computed by backpropagation and your estimate using finite difference should be 0.99 and 1.01.
* Do a gradient descent using mini-batch.
* Loss function:
  + -log(softmax(**o**)) where **o** is a vector of the outputs pre-activation of the neural network
* Empirical risk function: