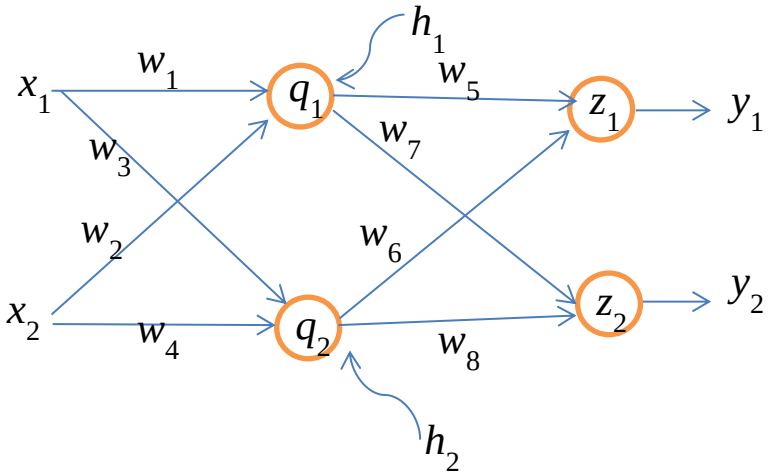
HW #6 Due: 6/12/2020

1. Justify the following claim: The back propagation algorithm does NOT work if weights in the neural network are all set to zero initially. To simplify the problem, use the following network and show that $\Delta w_1 = \eta \frac{\partial J}{\partial w_1} = 0$. The activation function from q_1 to h_1 and q_2 to h_2 is ReLU, the outputs y_1 and y_2 are softmax ouput, and the cost function is $J = -\log y_1$. Let w_1 to w_8 be all zeros, $x_1 = 0.5$, $x_2 = 0.5$, and $\eta = 0.1$.



2. We mentioned DCGAN and how to perform upsampling before. Consider the "CONV 1" stage in the generator (pp. 32 of the notes): One input channel has a size of 4×4, and one output channel has a size of 8×8 using a kernel of 5×5. It is understood that zeros are added before the convolution. Using the 1-D case as an example: Let the input sequence be x_1, x_2, x_3, x_4. Then, zero-padded input sequence would be $0,0,x_1,0,x_2,0,x_3,0,x_4,0,0,0$. With a filter of length 5, the resultant sequence after convolution has a length of 8. Based on this understanding, compute the minimal number of multiplications required to complete the "CONV 1" stage computation (i.e., most efficient way to obtain convolution results in software). Need not consider multiplications with biases.
3. In the version 1 model of the LSTM, the activation function for f_t is sigmoid (represented by σ in the notes). If we use ReLU instead, can the LSTM nodes still properly work? Why or why not?
4. Assuming that you are designing an anti-spam program based on neural networks with gradient descent. Though the training error seems acceptable, unfortunately the test error is too large to accept. What of the following

approaches are appropriate in your case? Explain.

- Try getting more training examples.
- Try a smaller set of features.
- Try a larger set of features.
- Run gradient descent with a different optimizer (such as RMSprop or ADAM).
- Try Newton's method.
- Use a smaller value for λ in the L2 norm term.
- Try using an SVM (different type of classifier).

5. The following is a simplified version of the variational autoencoder (VAE) to explain the reparameterization trick. In the figure, a node with "×" means multiplication, with "+" means addition, and with "exp" means $out = \exp(in)$. All other nodes use the sigmoid activation function. The activation function from q_i to s_i is sigmoid. In addition, ℓ_1 is a random number generated from a Gaussian random variable with zero mean and uni-variance. Let the loss function $\mathcal{L} = -(\mathcal{L}_1 + \mathcal{L}_2)$, where

$$\mathcal{L}_1 = \sum_{i=1}^{2}[x_i \ln y_i + (1 - x_i)\ln(1 - y_i)]$$

and

$$\mathcal{L}_2 = \frac{1}{2}\left(\mu^2 + \sigma^2 - \ln \sigma^2 - 1\right).$$

Find $\frac{\partial \mathcal{L}_1}{\partial z_1}$, $\frac{\partial \mathcal{L}_1}{\partial w_4}$, and $\frac{\partial \mathcal{L}_2}{\partial w_4}$