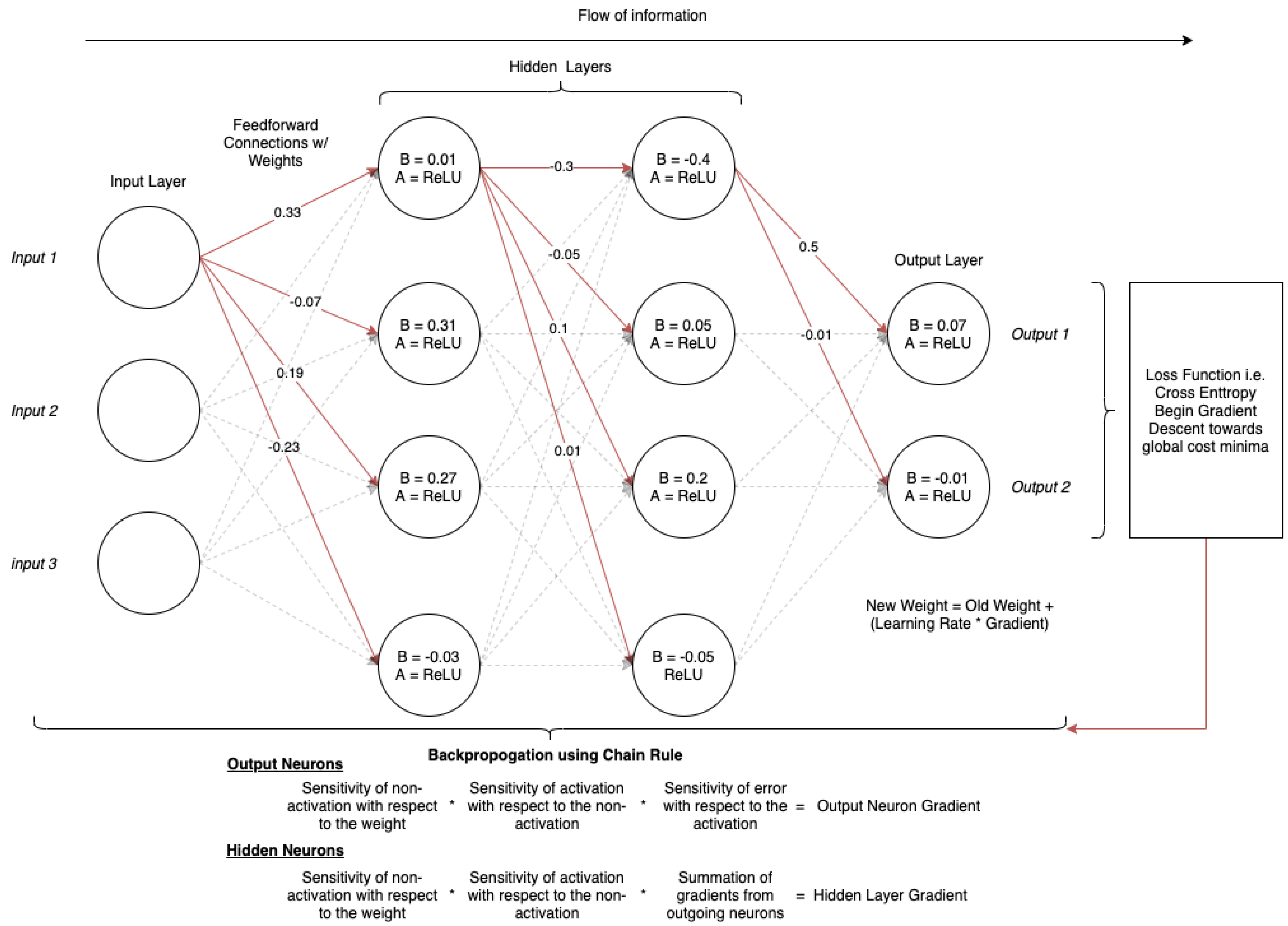
Feedforward Neural Network Architecture



1) ***Weights are initialised*** i.e. Random, Normal Distribution, Xavier, He Norm

2) The Forward Propagation occurs
- Computing the non-activation of each neuron
- Non-activation = Summation of (inputs multiplied by weight) + bias
- Computing the activation of each neuron

3) ***The Loss Function calculates the error*** between actual and predicted values

4) ***Using gradient descent, backpropagation occurs***, whereby the objective is to minimise the error by adjusting the weights to reach global minima.
- The amount of times the weights are adjusted depends on minibatch size
- And number of epochs
- Weight Update Rule: New Weight = Old Weight – Learning Rate * Gradient

5) Calculating the gradient for Output Layer Neurons

$$\frac{\partial \mathcal{E}_k}{\partial w_{hi}^k} = \frac{\partial \mathcal{E}_k}{\partial \mathcal{S}(y_j^k)} \frac{\partial \mathcal{S}(y_j^k)}{\partial y_j^k} \frac{\partial y_j^k}{\partial w_{hi}^k}$$

- Requires derivative of Error Function and derivative of Activation Function

6) Calculating the gradient for Hidden Layer Neurons

$$\frac{\partial \mathcal{E}_k}{\partial w_{ih}^k} = \sum_{j=1}^{p} \left\{ \frac{\partial \mathcal{E}_k}{\partial \mathcal{S}(y_j^k)} \frac{\partial \mathcal{S}(y_j^k)}{\partial y_j^k} \frac{\partial y_j^k}{\partial \mathcal{S}(z_h^k)} \right\} \mathcal{S}'(z_h^k)\, \mathcal{S}(x_i^k)$$