1) $Weighted\ Sum\ of\ neuron = w_0 a_0 + w_1 a_1 + ... + w_n a_n + b$

- Calculates the weighted sum of a neuron in a hidden or output layer
- w_i = A weight to this neuron from one in the previous layer
- a_i = The activation of a neuron in the previous layer
- b = The bias of this neuron

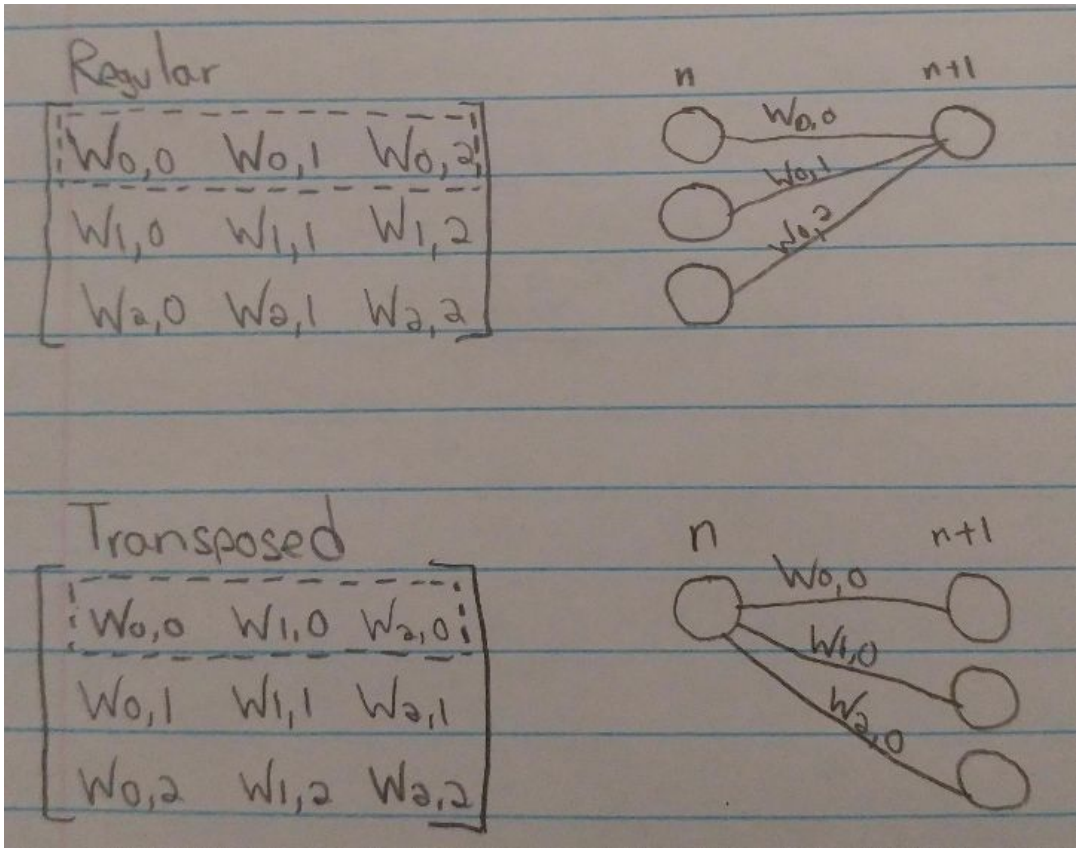
2) $Activation\ of\ neuron = \sigma(Weighted\ Sum)$

- Normalizes the activation (output) of a neuron to a value between [0.0 - 1.0]
- σ = The sigmoid activation function

3) $Error\ of\ neuron\ (output\ layer) = \delta_j^L = (a_j - y_j) * \sigma'(z_j^L)$

- Calculates the error of neuron *j* in the last layer. The error of a neuron tells us how far off the neuron's actual output was from the target output. This value is used for backpropagation.
- δ_j^L = The error δ of neuron j in the last layer (L)
- a_j = The activation of neuron j in the last layer
- y_j = The target output for neuron j in the last layer
- $\sigma'(z_j^L)$ = The sigmoid prime of the weighted sum of neuron j in the last layer

4) $Error\ of\ neurons\ (input/hidden\ layer) = \delta^n = ((w^{n+1})^T * \delta^{n+1}) \odot \sigma'(z^n)$

- Calculates the errors of neurons in input and hidden layers. The errors in these layers depend on the errors in the layers after them, this is why we start with the output layer and move backwards through the network for backpropagation.
- δ^n = A vector representing the errors of neurons in layer n
- $(w^{n+1})^T$ = The transpose of the weight matrix between layer n and layer n+1. This will allow us to multiply the vector of errors in the next layer with their corresponding weights. After transposing, each row will represent weights to one neuron in layer n+1. The reason this works is better shown below:
- 

- δ^{n+1} = A vector representing the errors of neurons in layer n + 1
- $\odot$ = Element-wise multiplication
- $\sigma'(z^n)$ = The sigmoid prime of the weighted sum (z) of neuron n. This is how fast the sigmoid function is changing with respect to z^n.

5) $Gradient\ of\ any\ bias = \delta_j^n$

- Lets us calculate the gradient of any bias in the network. This value tells us how much we should adjust a given bias in order to improve the network's accuracy. The bias of neuron j in layer n is equal to the error of neuron j in layer n. Simply put, the gradient of the bias of a neuron is equal to the error of that neuron.

6) $Gradient\ of\ any\ weight\ (w_{jk}^n) = a_k^{n-1} * \delta_j^n$

- This value tells us how much we should adjust a given weight in order to improve the network's accuracy. The gradient of any weight in the network is equal to the activation of the neuron in the previous layer multiplied by the error of the neuron in this layer.
- w_{jk}^n = The weight from neuron k in layer n to neuron j in layer n+1.
- a_k^{n-1} = The activation of neuron k in layer n-1
- δ_j^n = The error of neuron j in layer n