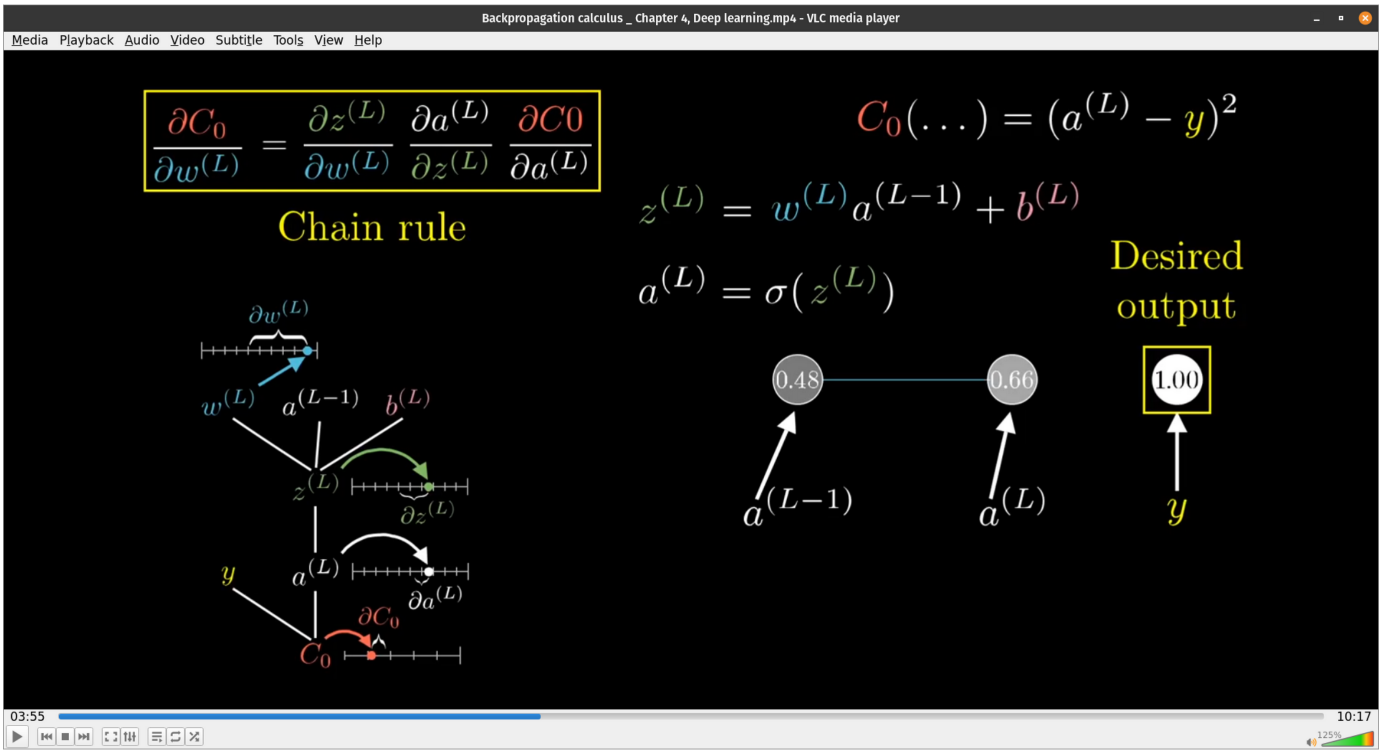
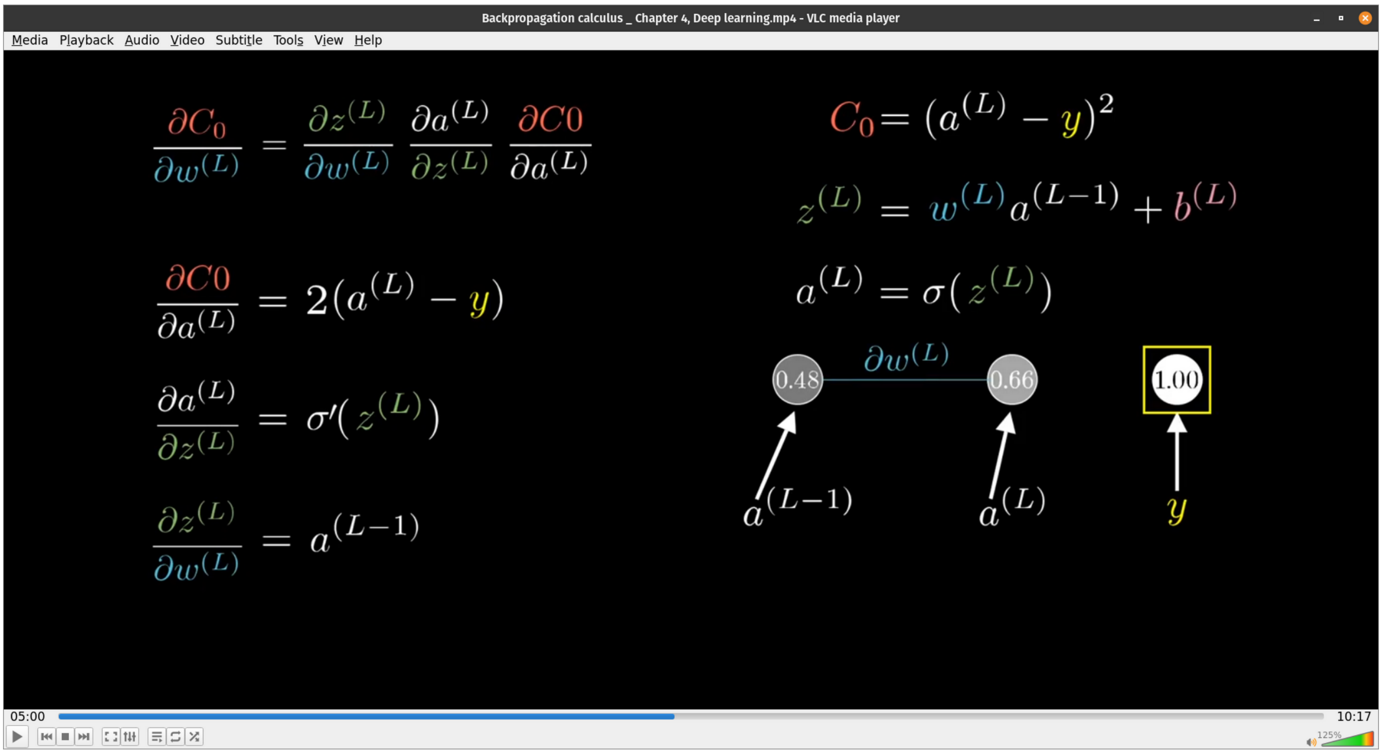
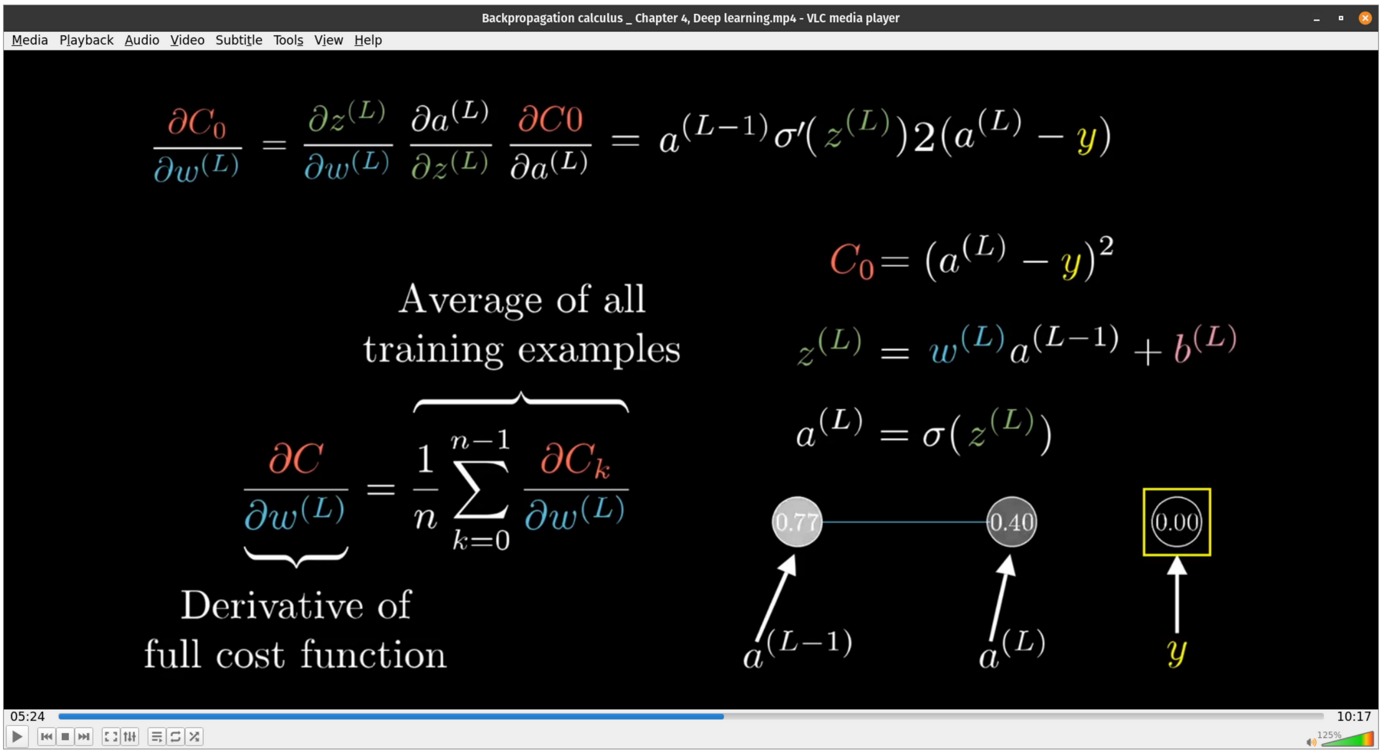
Backpropagation: Calculus



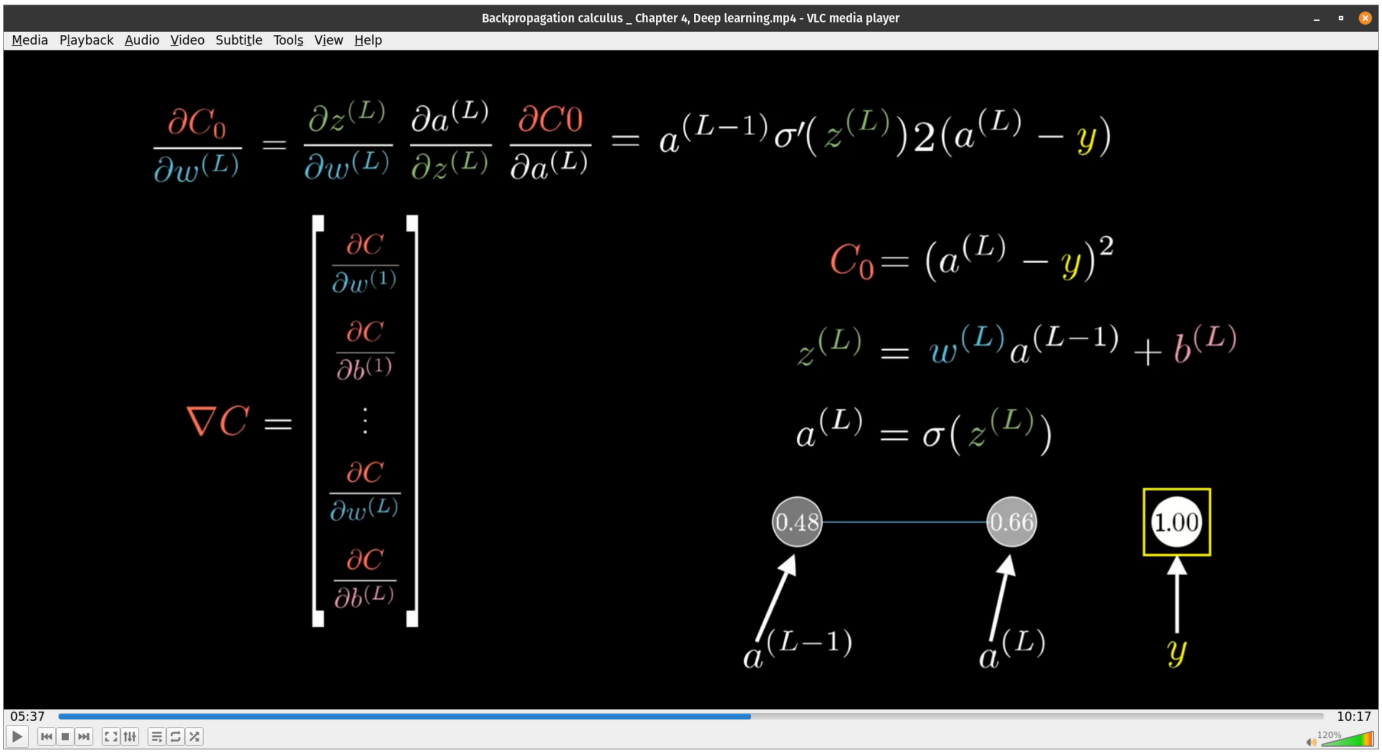
This flowchart shows how the change in weight affect the change in $z^{(L)}$ which affects the change in $a^{(L)}$ which finally affects the change in cost C_0

σ is the sigmoid activation function, which is not used much in the present times, rather $Tanh$ or $ReLU$ is used due to slow convergence of Sigmoid and 0 centric base in Tanh and finally, fast computation of ReLU.





This expression is just one component of the gradient vector which is built up from the partial deriavtives of all those weights and biases.



sensitivity w.r.t. **bias** is almost idenical by replacing the $\partial w^{(L)}$ by $b^{(L}$ whose derivative is = `1`.

So, the final expression for sensitivity w.r.t. bias becomes:

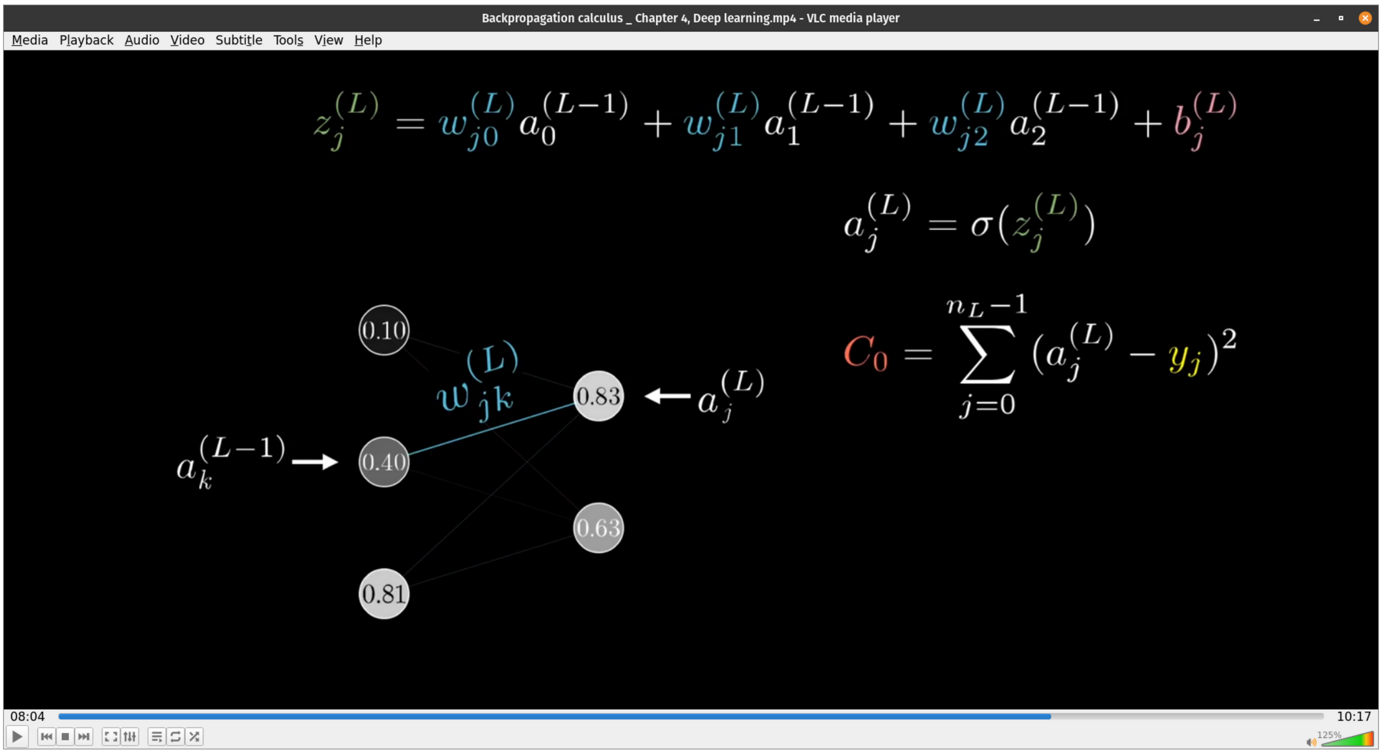
$$\frac{\partial C_0}{\partial w^{(L)}} = \sigma'(z^{(L)})2(a^{(L)} - y)$$

NOTE

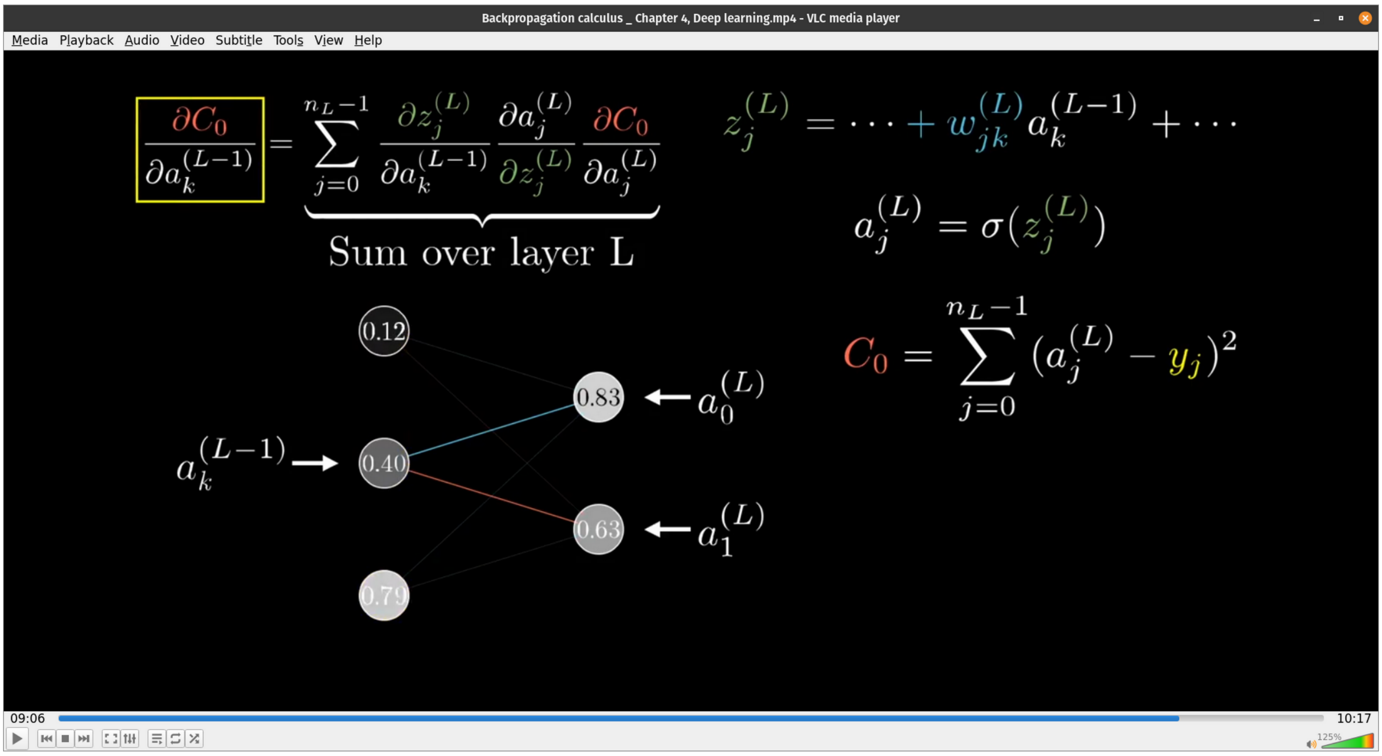
The calculations above are for a model which has a single neuron in all of its layers, though the above work is quite similar for a more complex neural network structure.

Calculations for more complex neural network

Notations:



Final Equations:



Notice the slight change... instead of taking derivative of cost w.r.t. weight, we took it w.r.t. the `k`th activation in the layer `L-1`

n_L represents the number of neurons in the layer `L`.