Neural Network Basics

João Campagnolo

Computational Neuroscience class

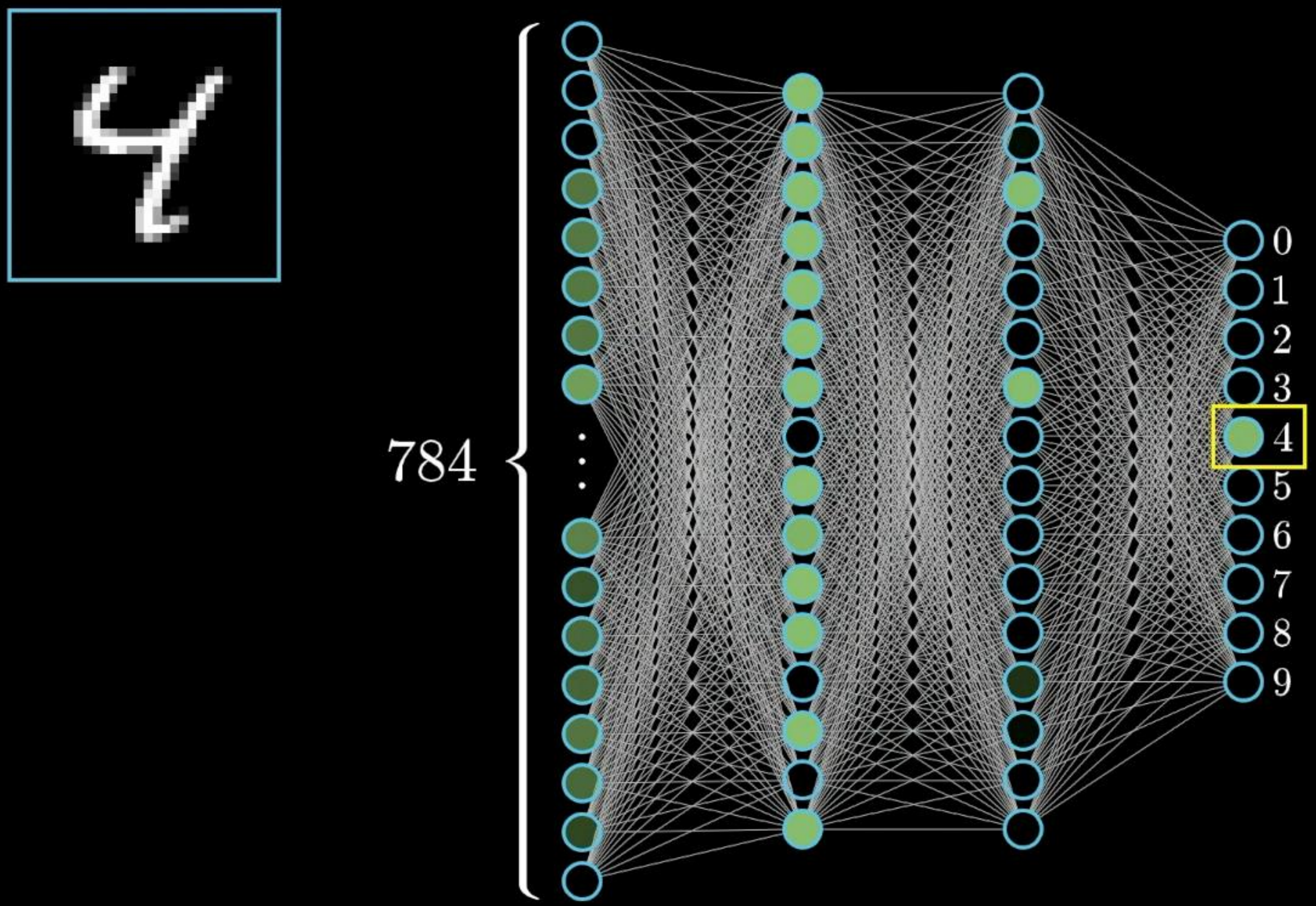
Spring 2024

UNIVERSITY OF COPENHAGEN

Let’s start with the example from [3b1b]

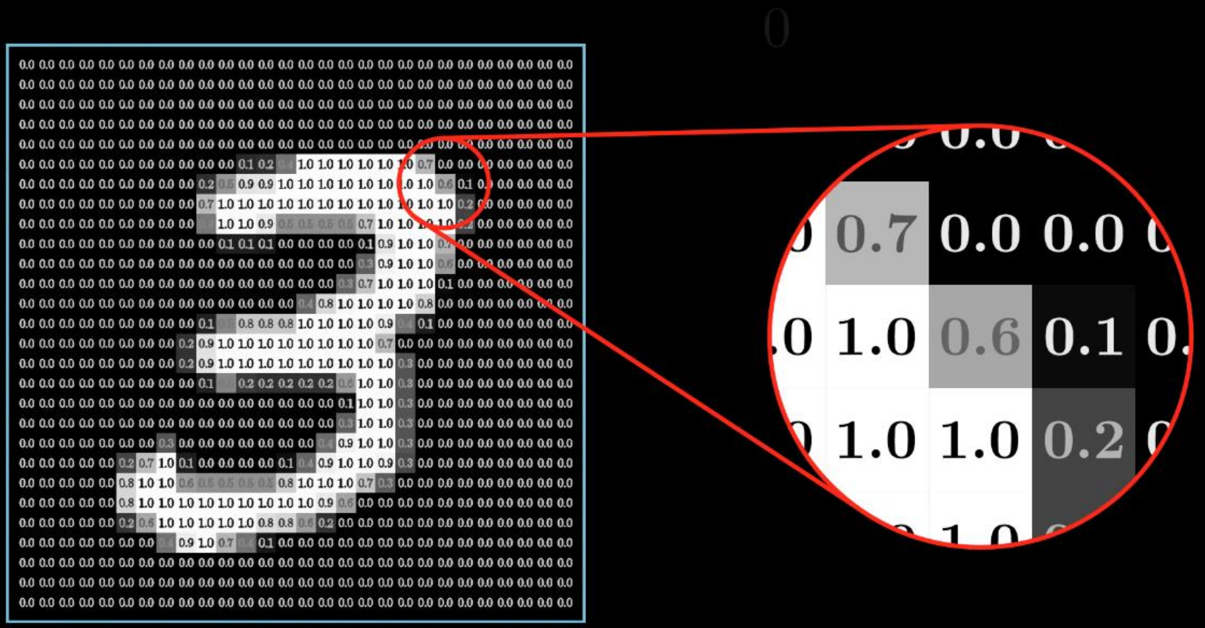
Plain vanilla

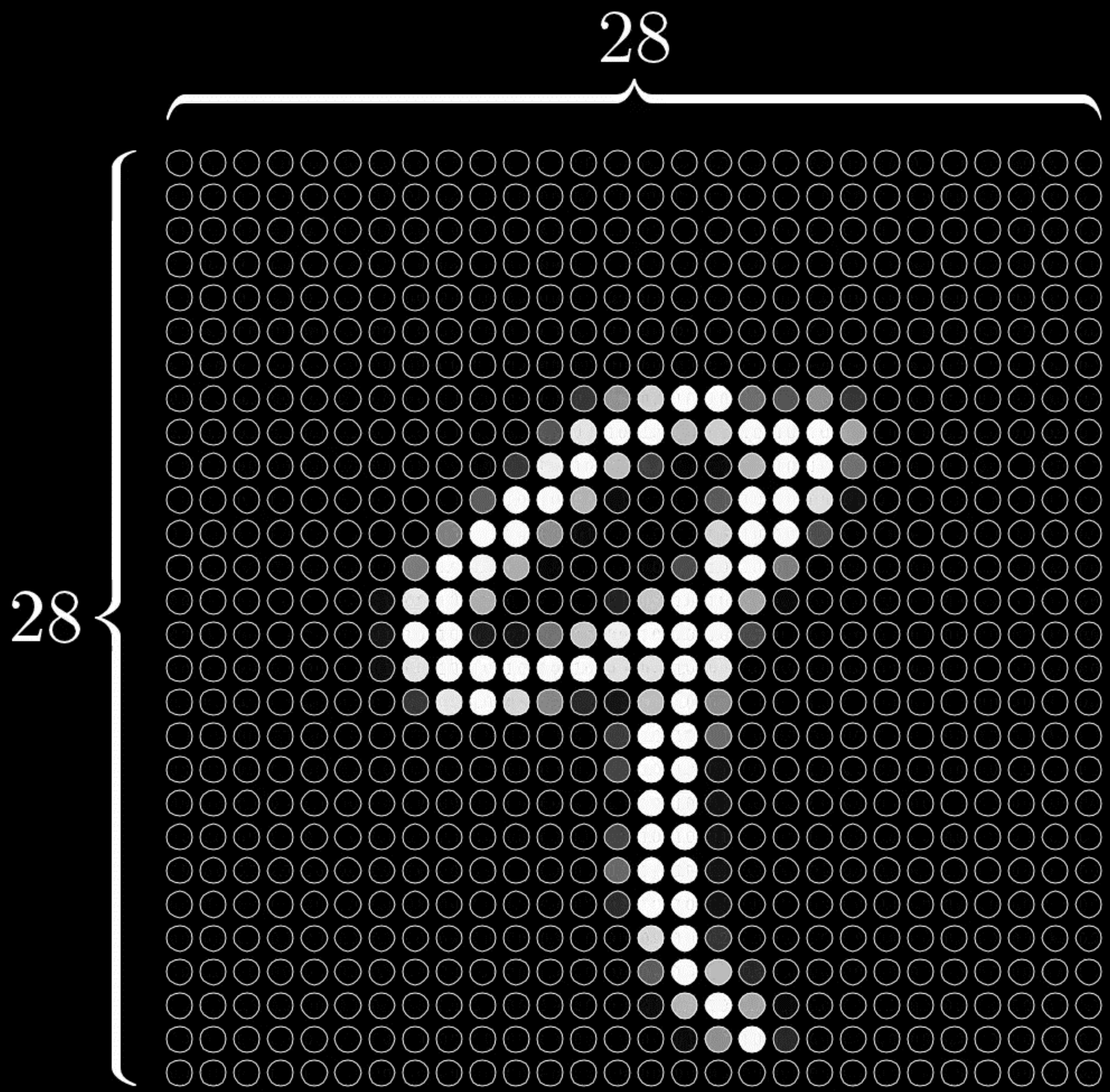
(aka "multilayer perceptron")



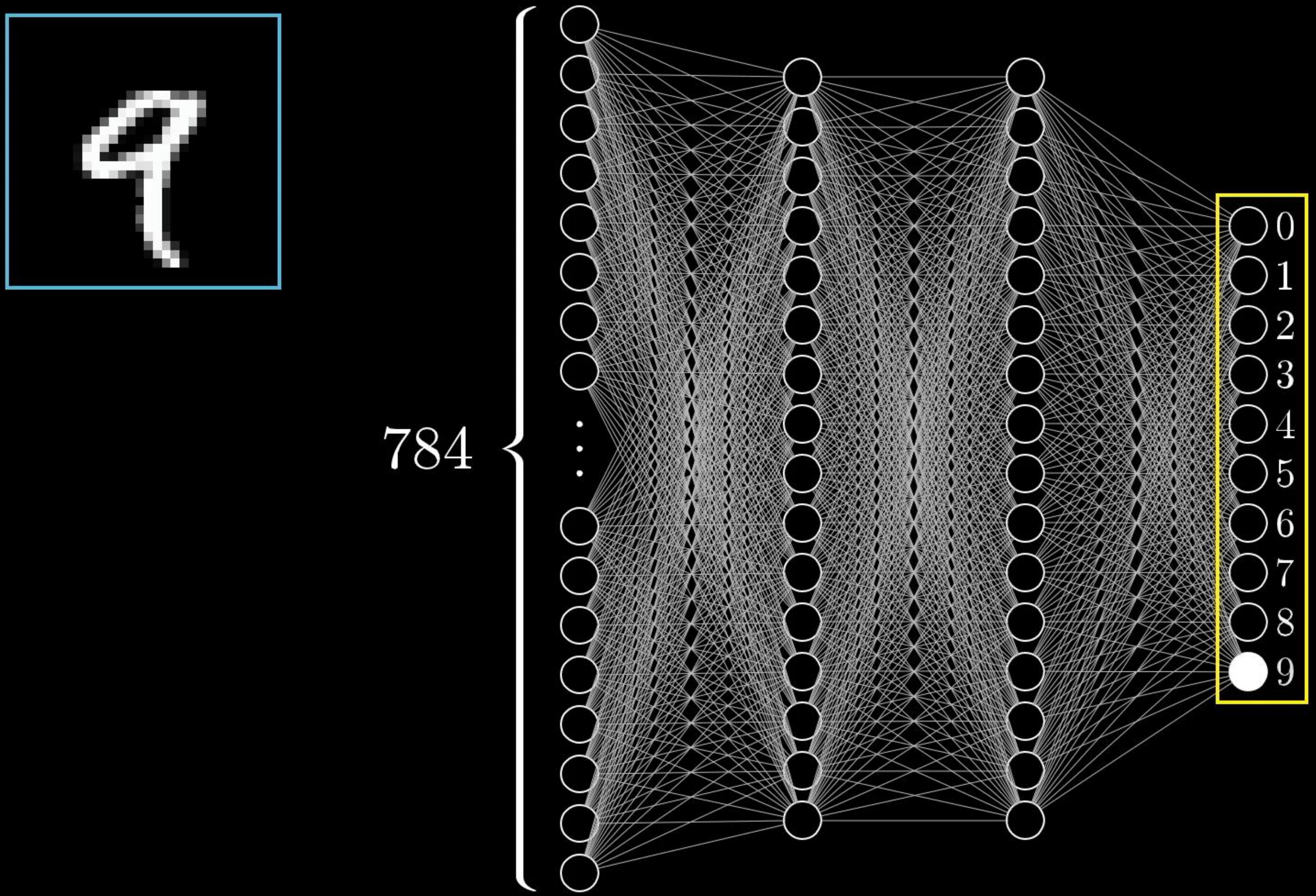
Every neuron has an activation between 0.0 and 1.0, sort of analogous to how neurons in the brain can be active or inactive.

Each pixel in the original image has a value between 0.0 (black) and 1.0 (white).

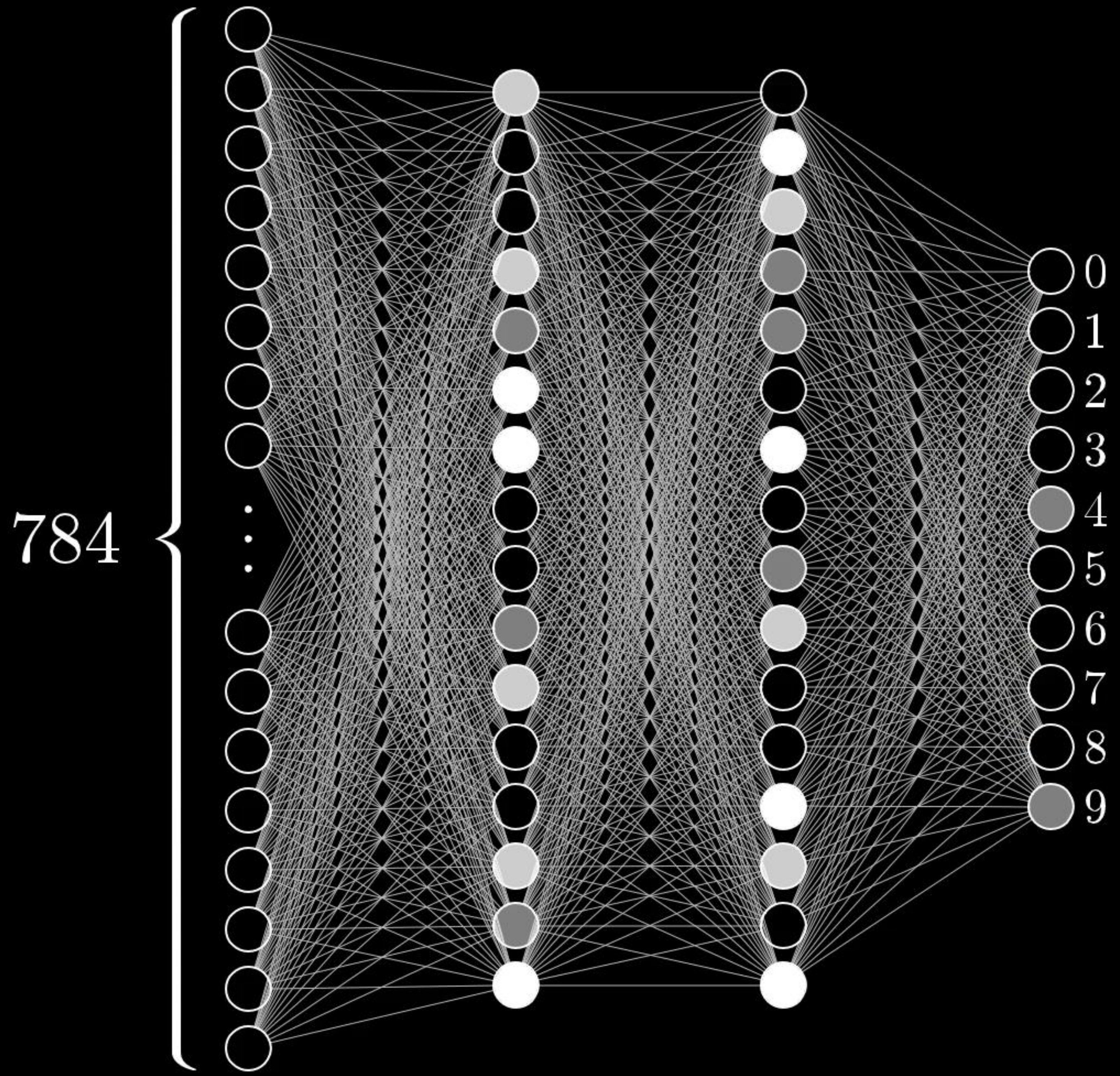




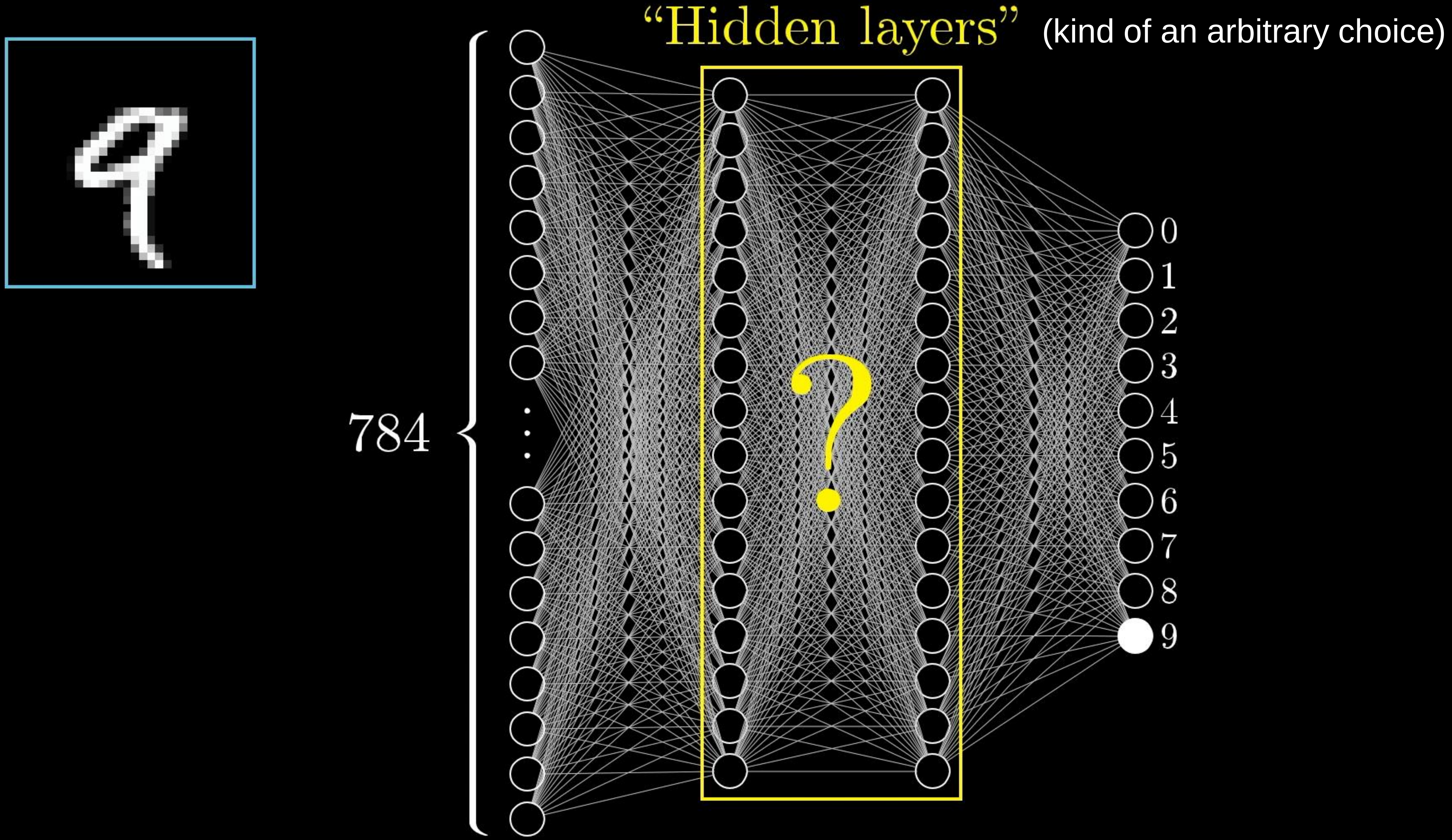
$$28 \times 28 = 784$$



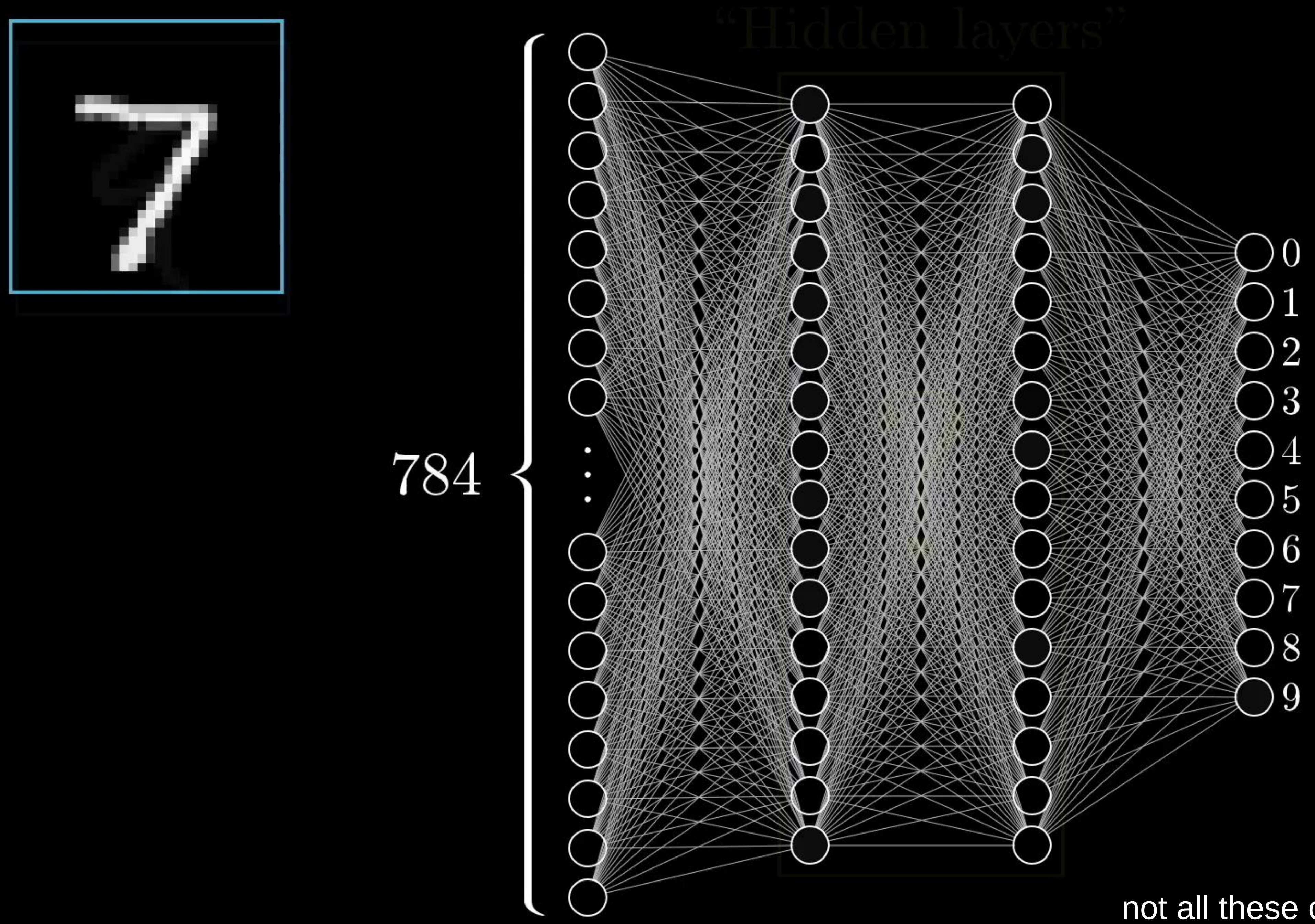
Set input and output dimensions



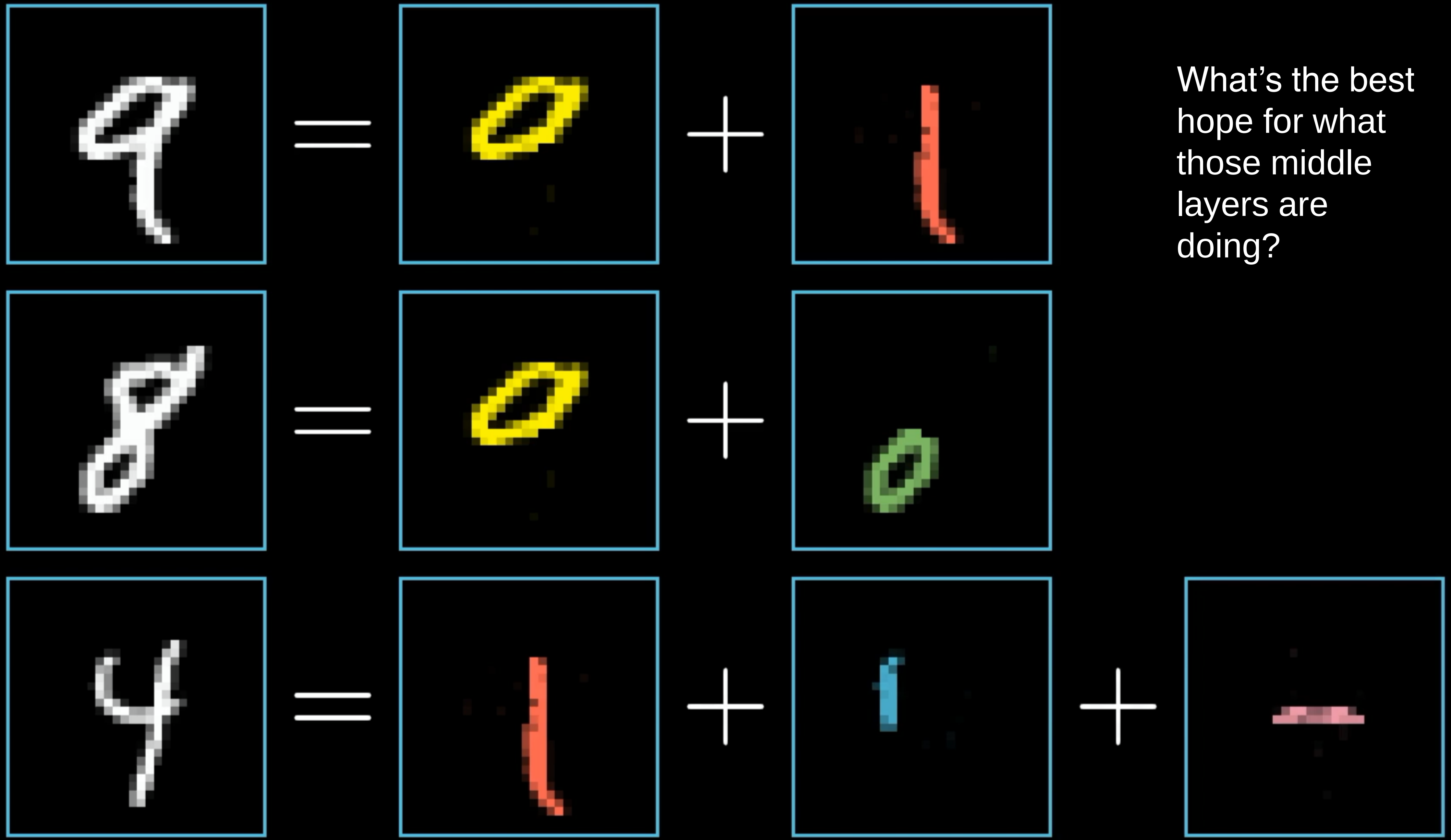
What is the output the network will choose?



“Hidden layers”
(kind of an arbitrary choice)
784
?
0
1
2
3
4
5
6
7
8
9

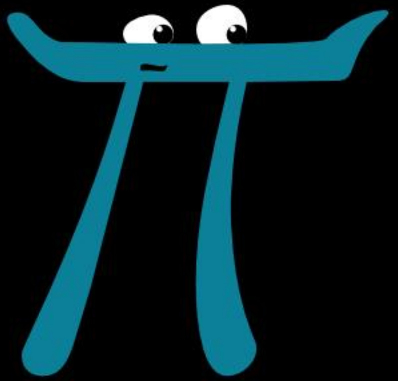


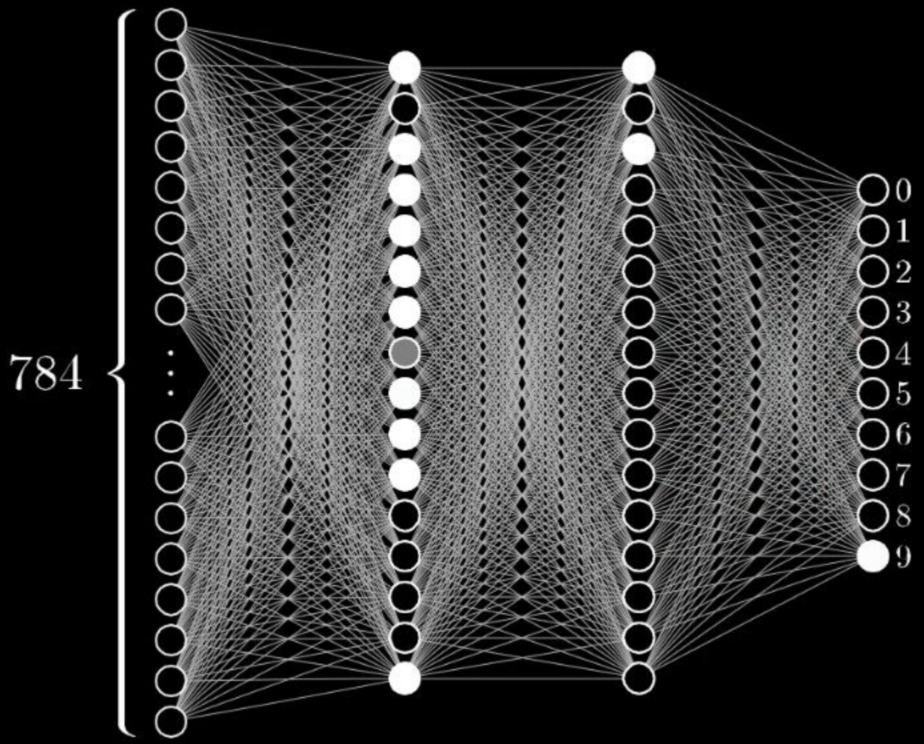
not all these connections are equal

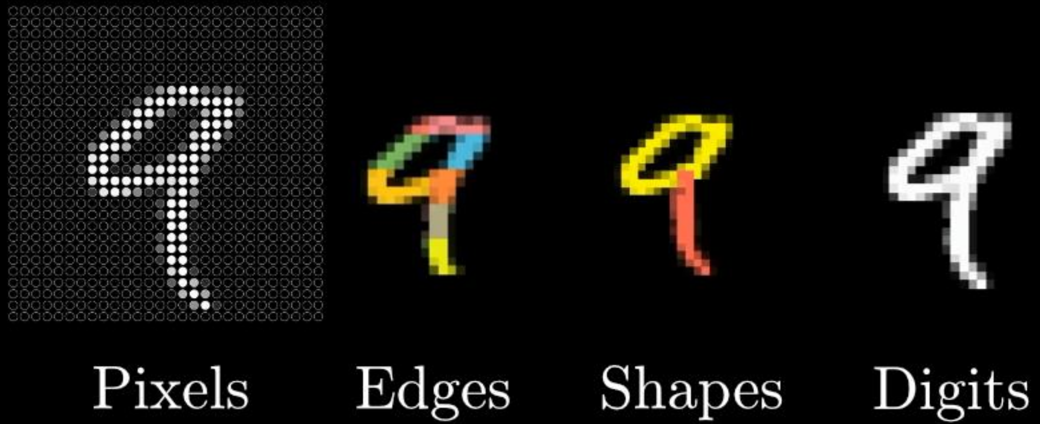


What’s the best hope for what those middle layers are doing?

Does the network actually do this?



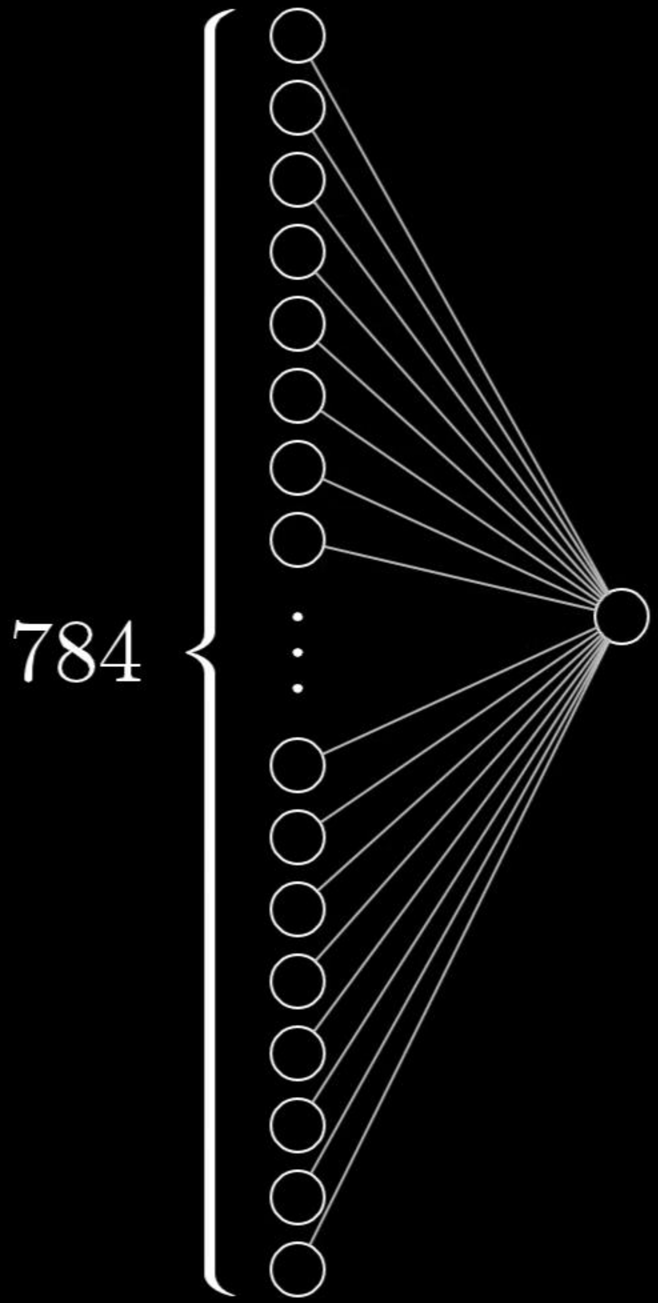


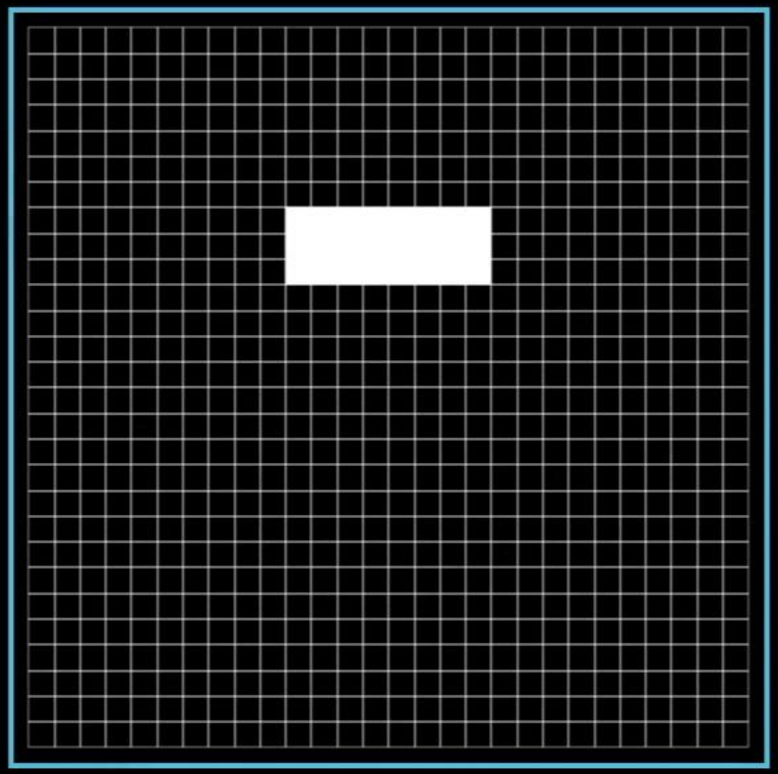


Could each neuron in the second layer of the network corresponds to some little edge?

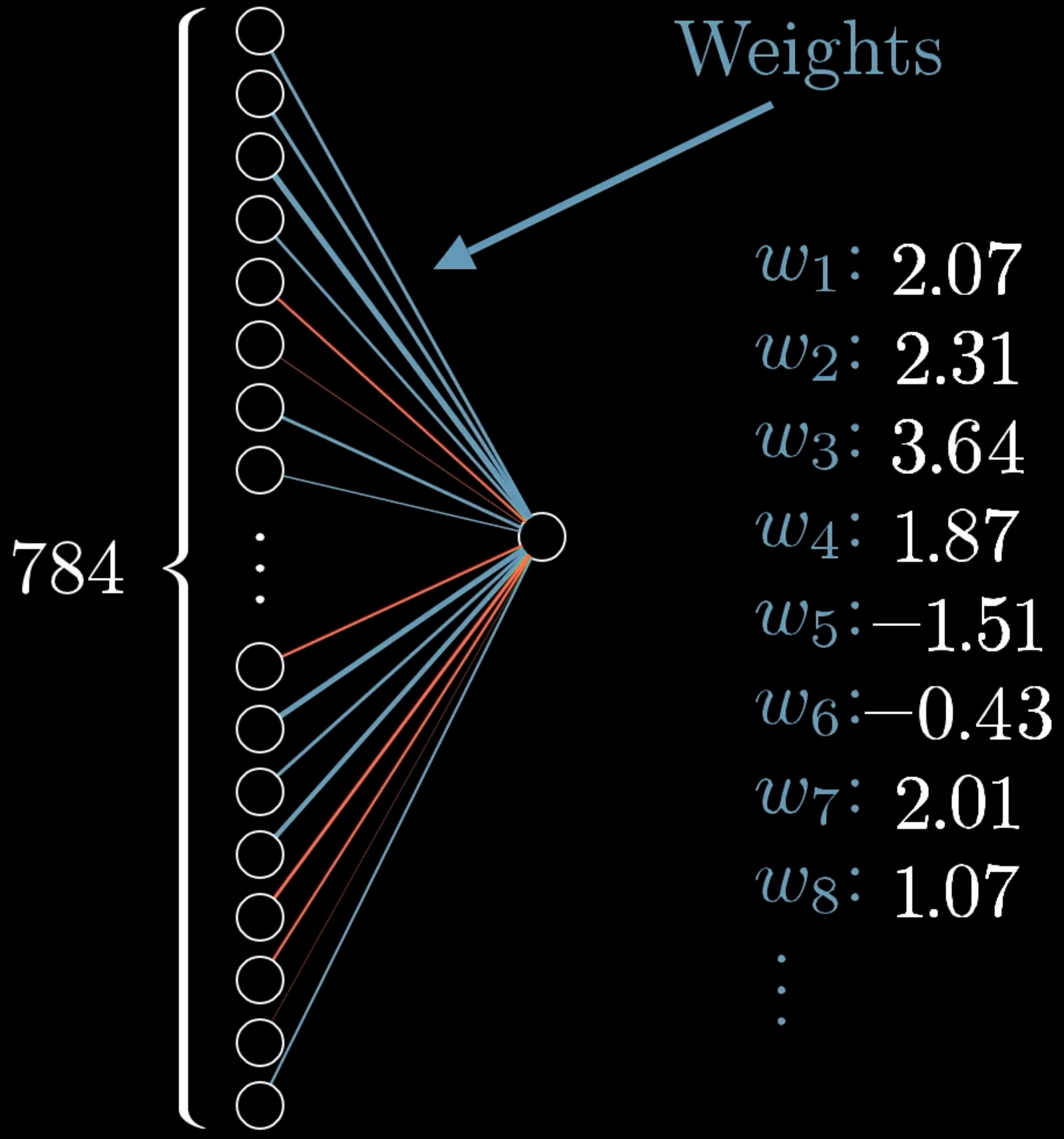
“The layered structure of the neural network is great because it allows you to break down difficult problems into bite-size steps, so that moving from one layer to the next is relatively straightforward*.”*

How Information Passes Between Layers

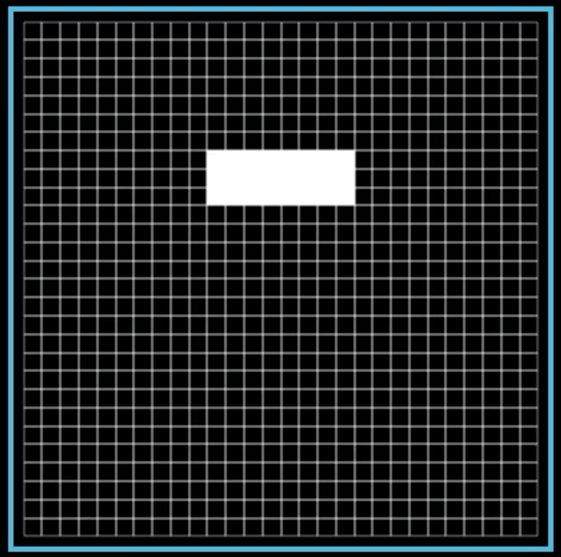




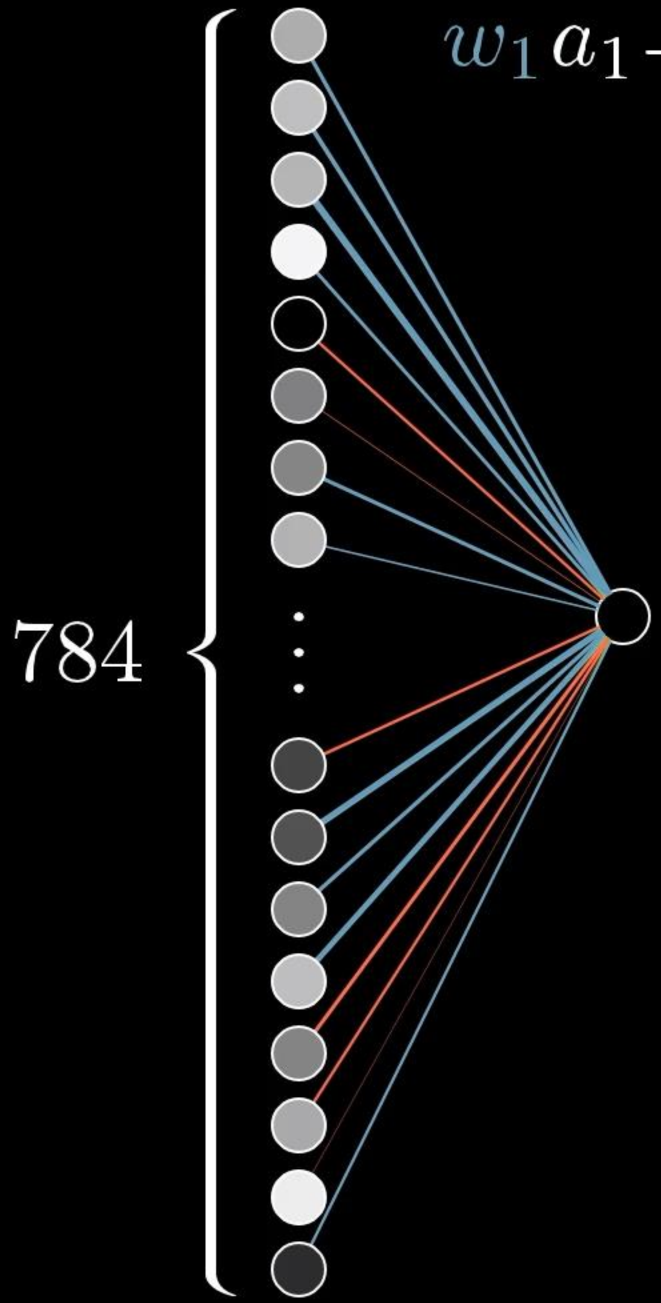
We want this one, specific neuron in the second layer to pick up on whether the image contains this one, specific edge.



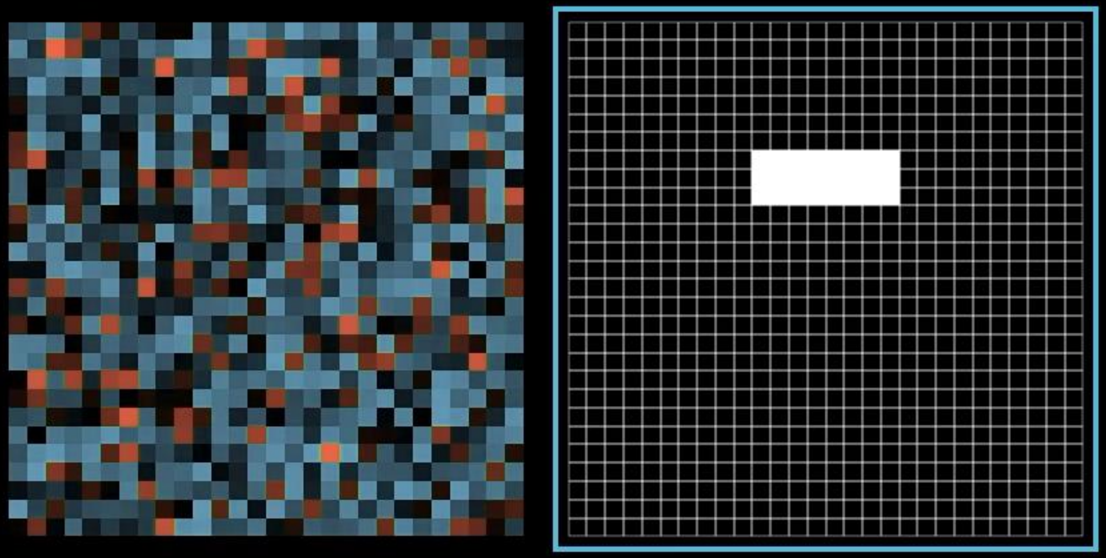
Each weight is an indication of how its neuron in the first layer is correlated with this new neuron in the second layer.

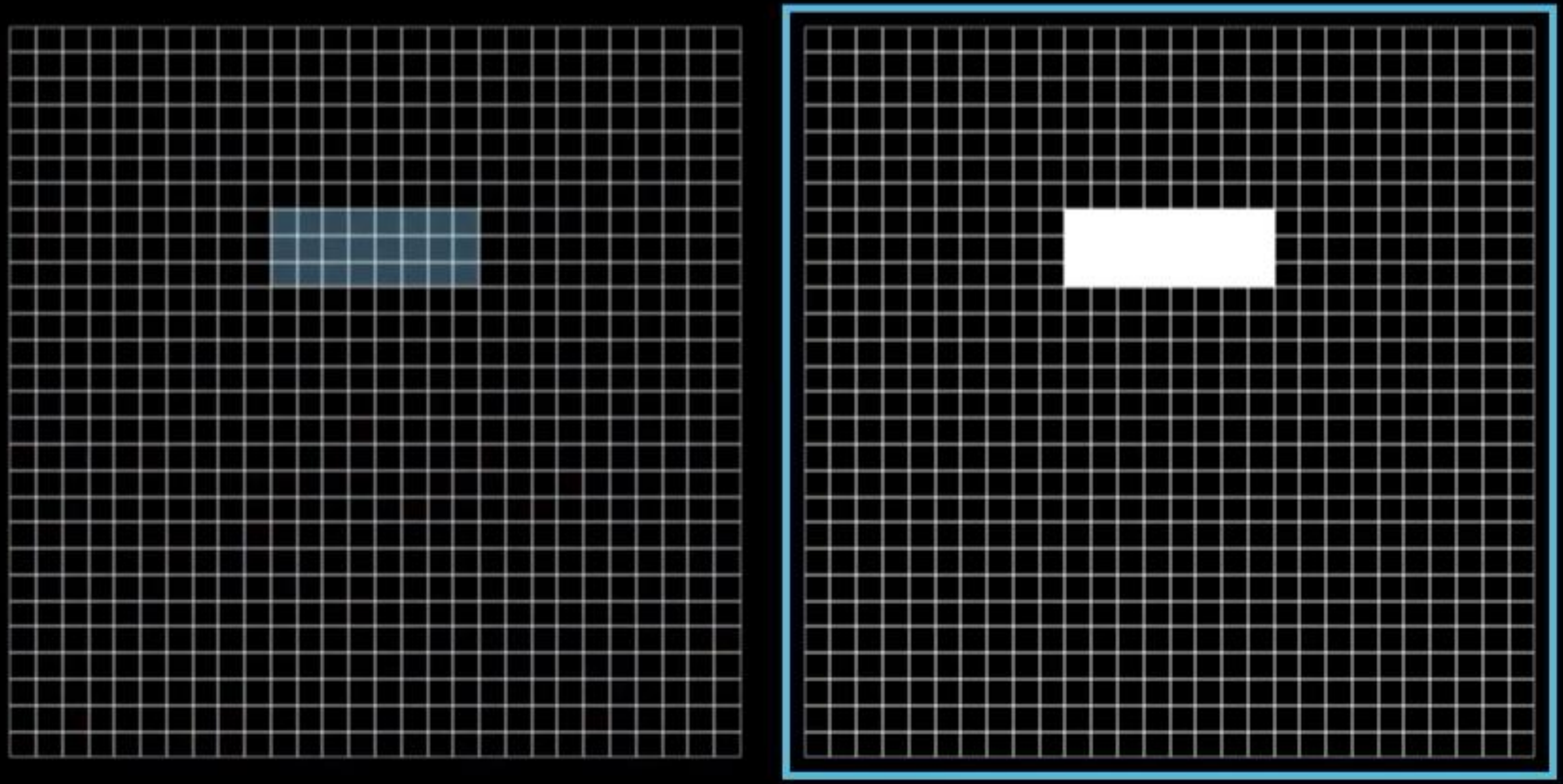


$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \cdots + w_n a_n$$

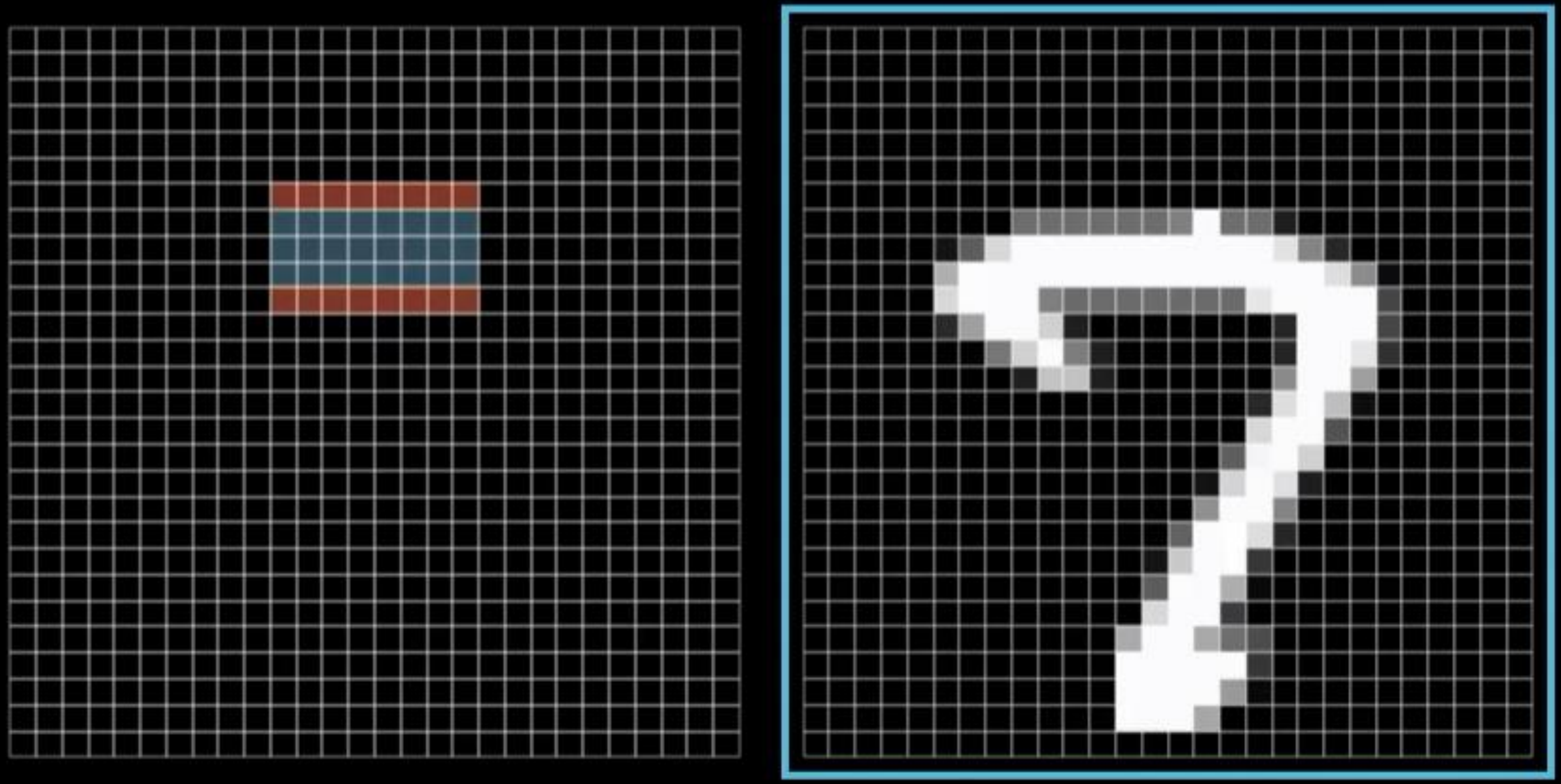


Arranging the weights into this 28x28 grid:

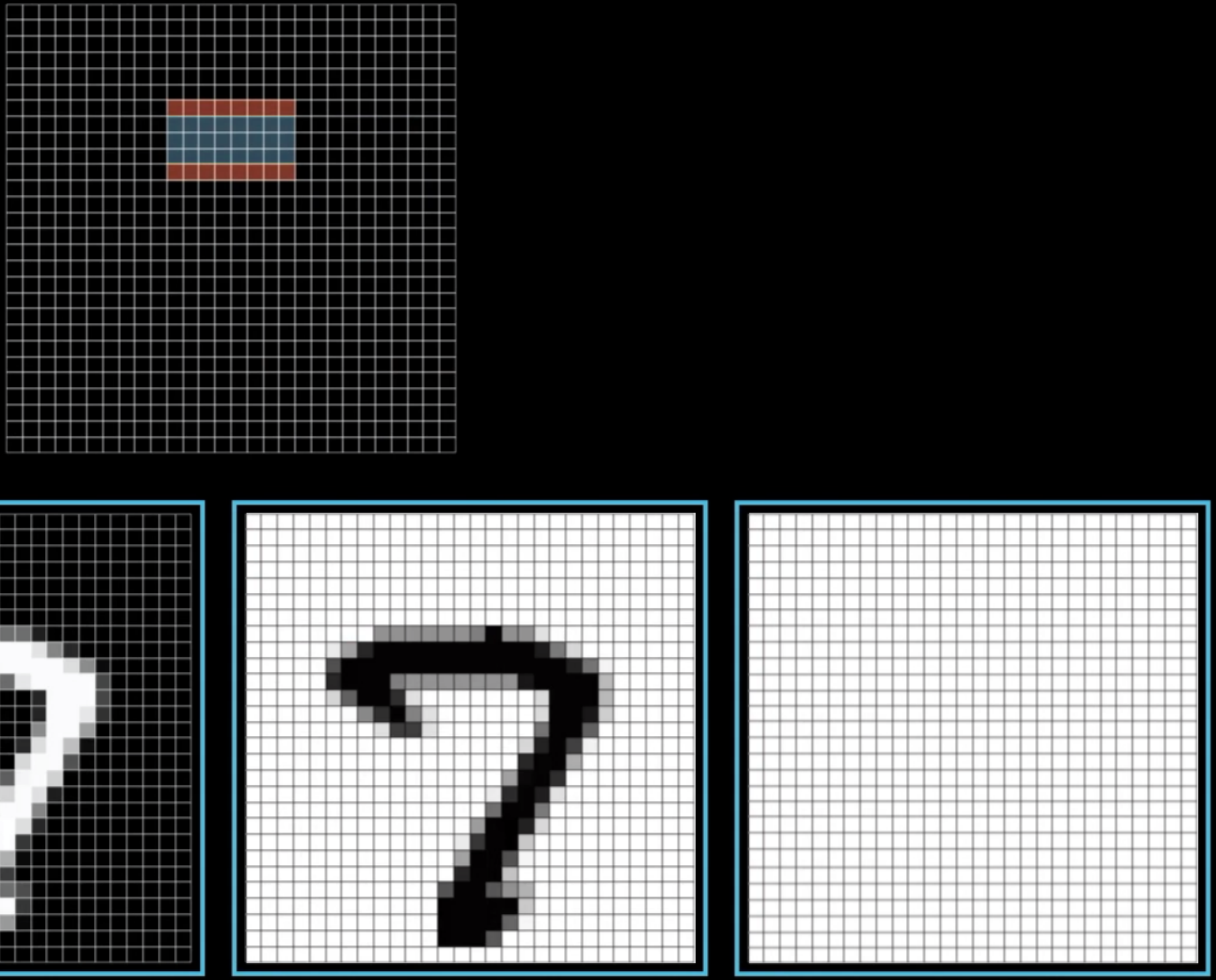




With these weights, the neuron in the second layer will be more activated when pixels in this region are more activated.

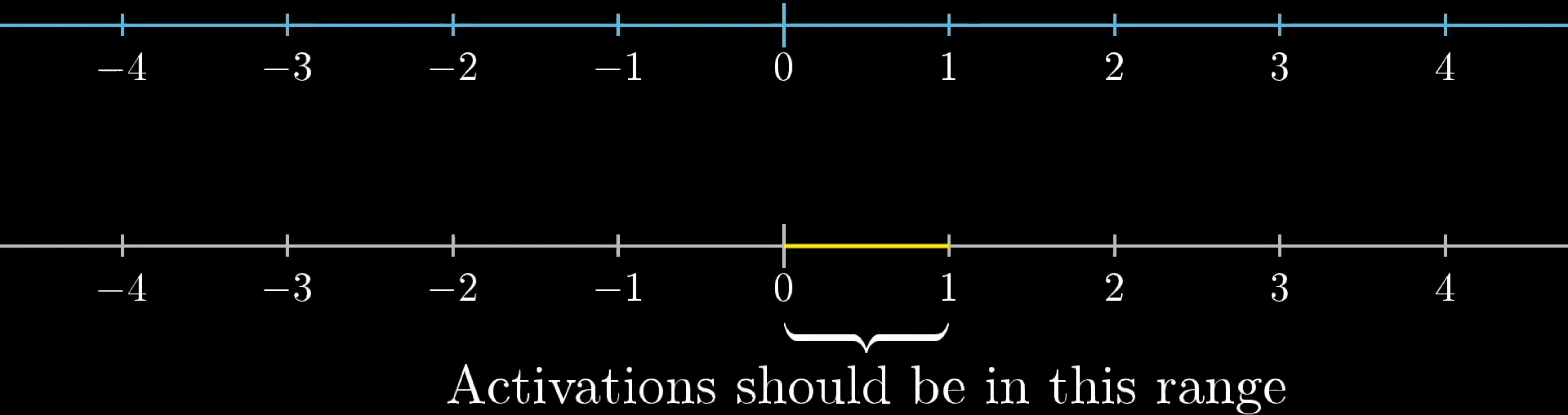


To really pick up on whether or not this is an edge, you might want to have some negative weights associated with the surrounding pixels.

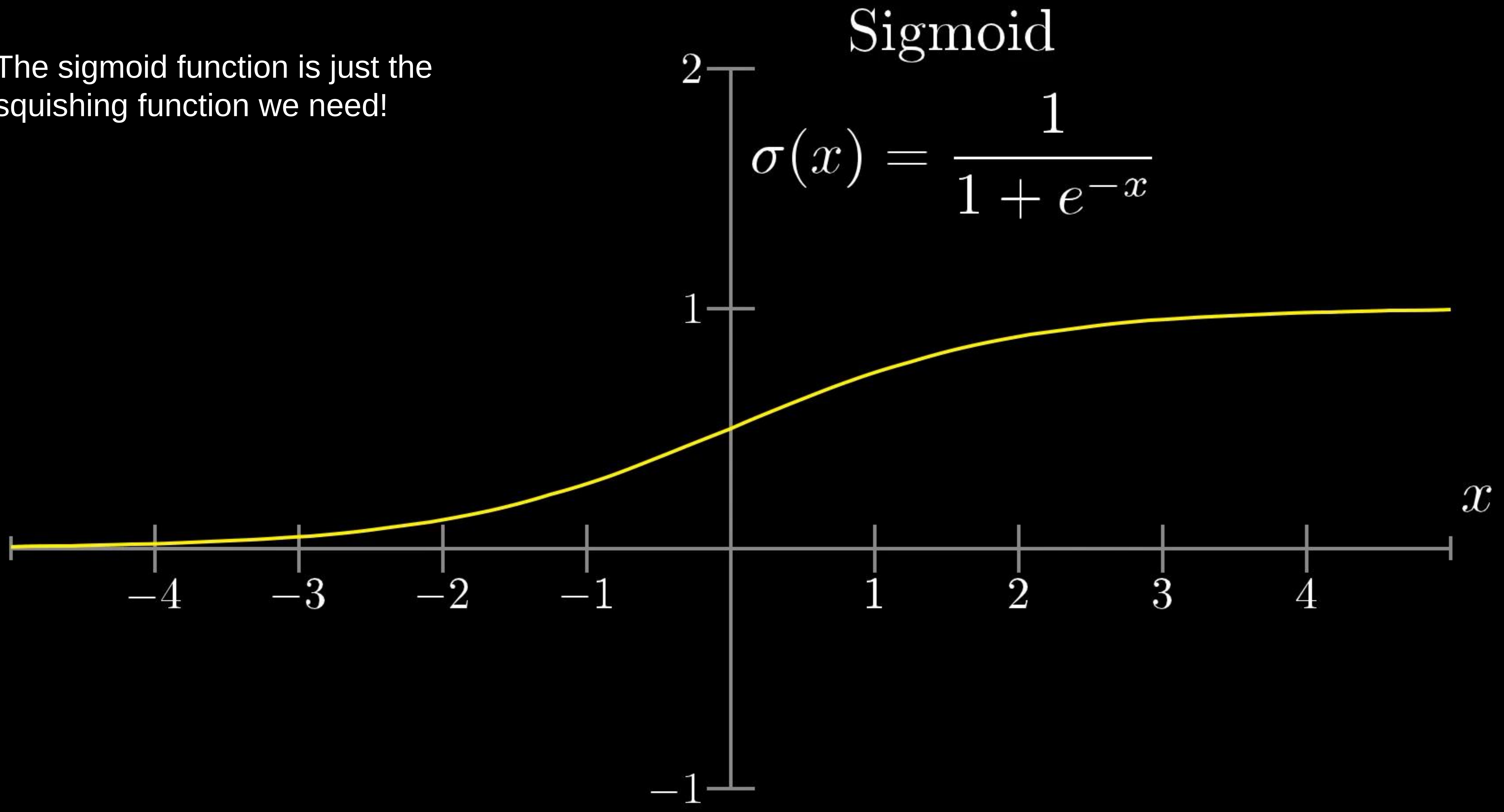


A B C D

$$w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4 + \cdots + w_n a_n$$



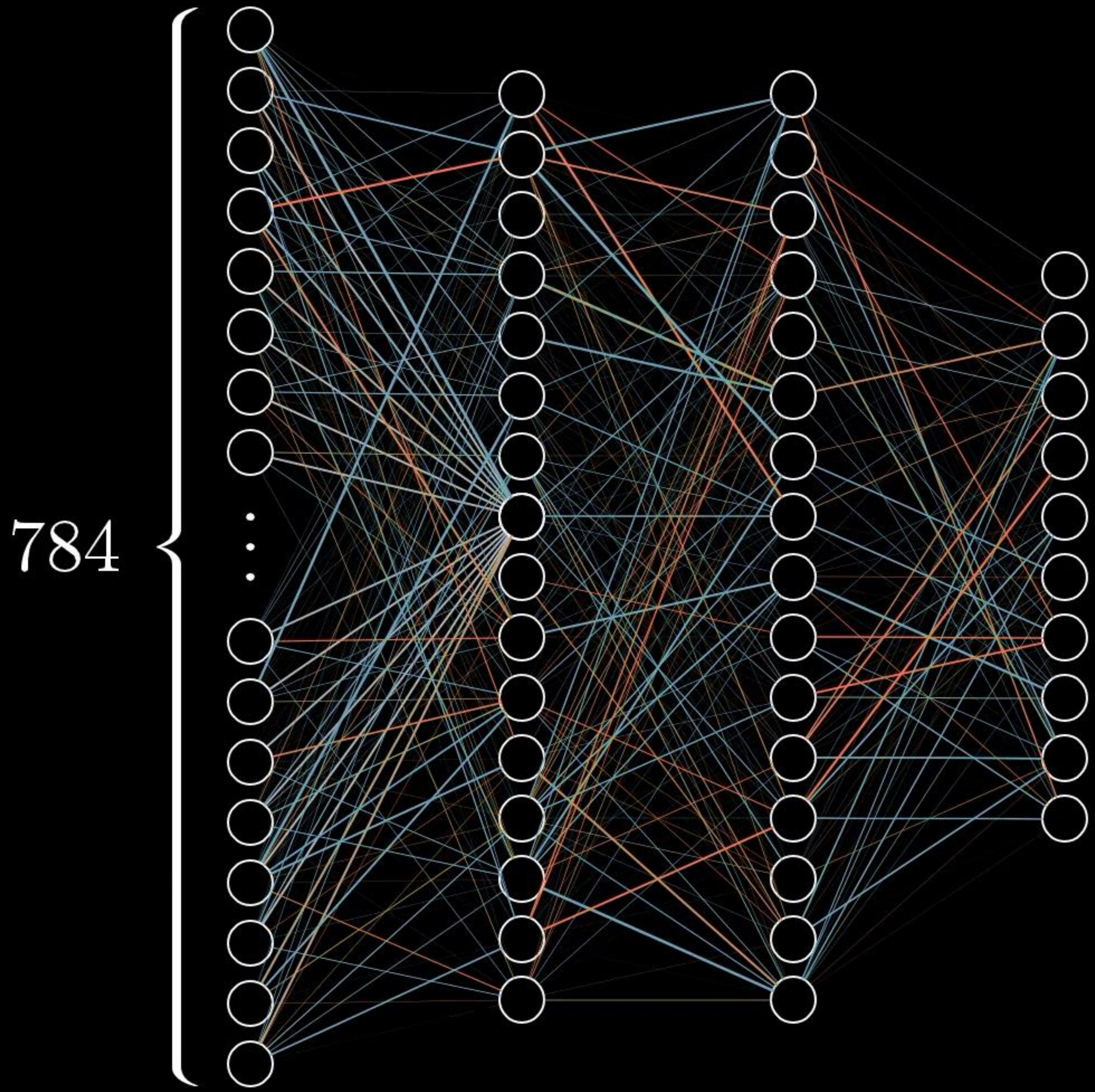
The sigmoid function is just the
squishing function we need!



Maybe we only want it to be meaningfully active when that sum is bigger than, say, 10.

$$\sigma\left(w_1 a_1 + w_2 a_2 + w_3 a_3 + \cdots + w_n a_n \underbrace{\boxed{-10}}_{\text{“bias”}}\right)$$

Only activate meaningfully
when weighted sum > 10



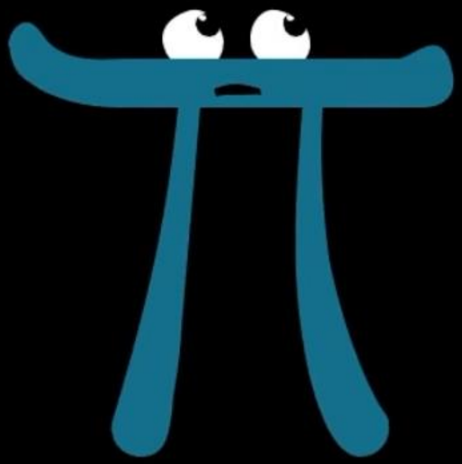
$$784\times16+16\times16+16\times10$$

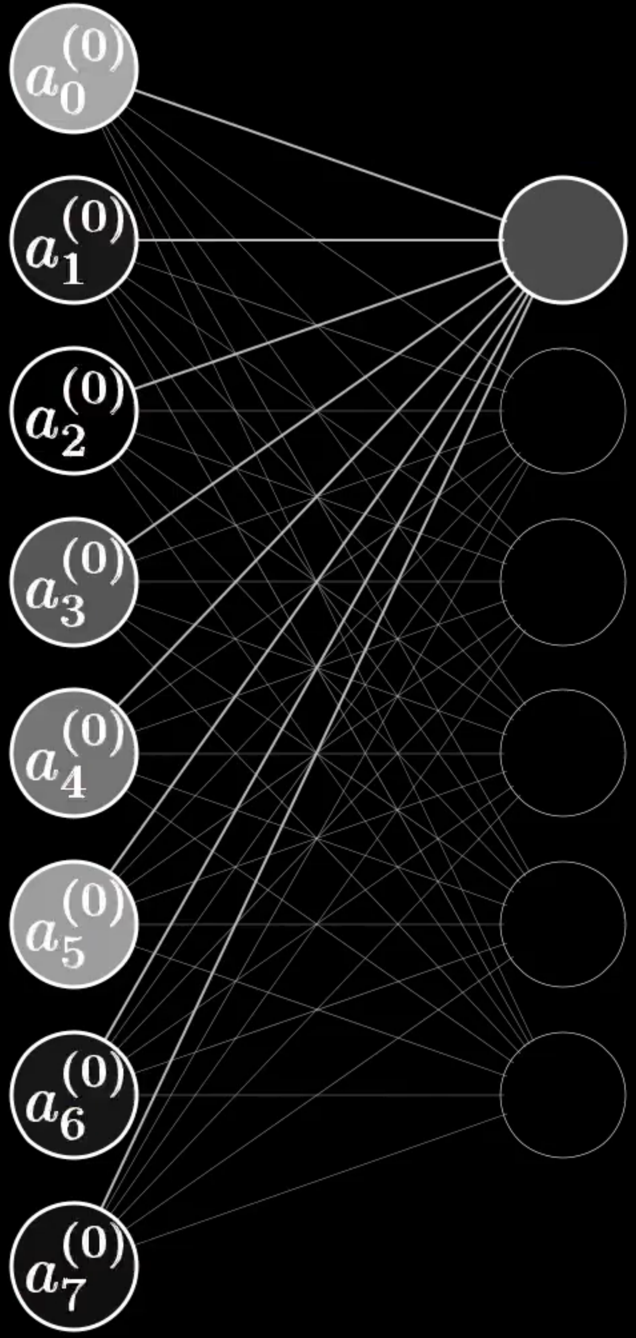
weights

$$16+16+10$$

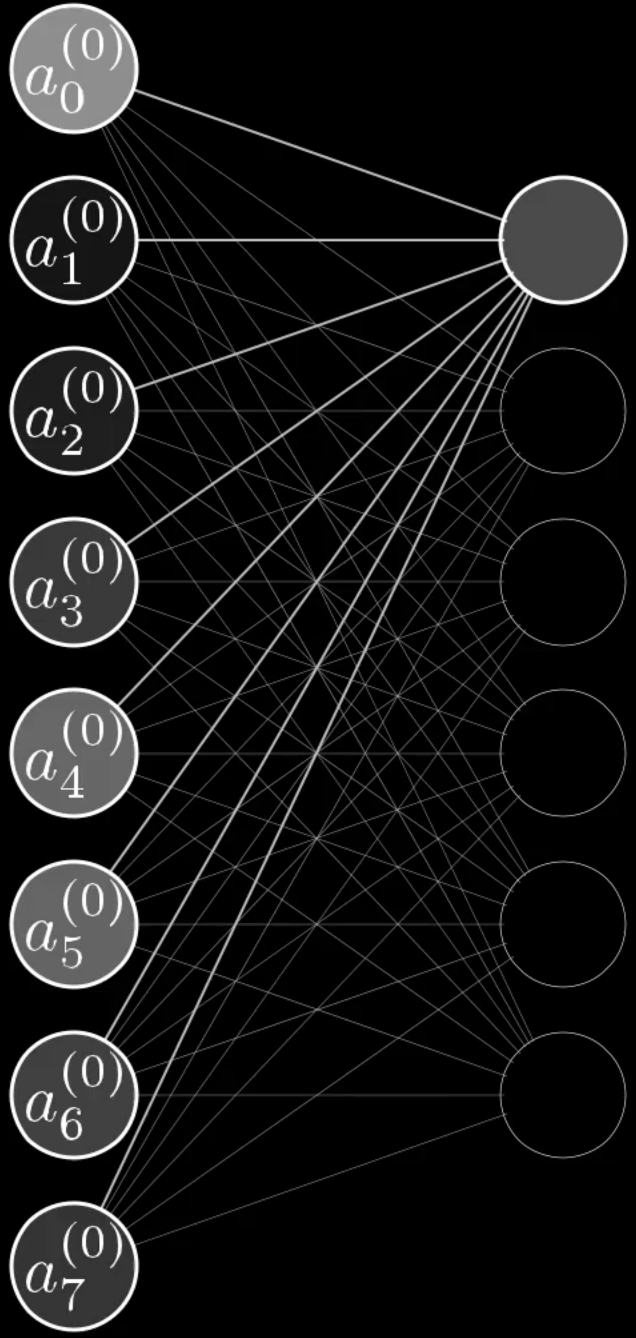
biases

13,002



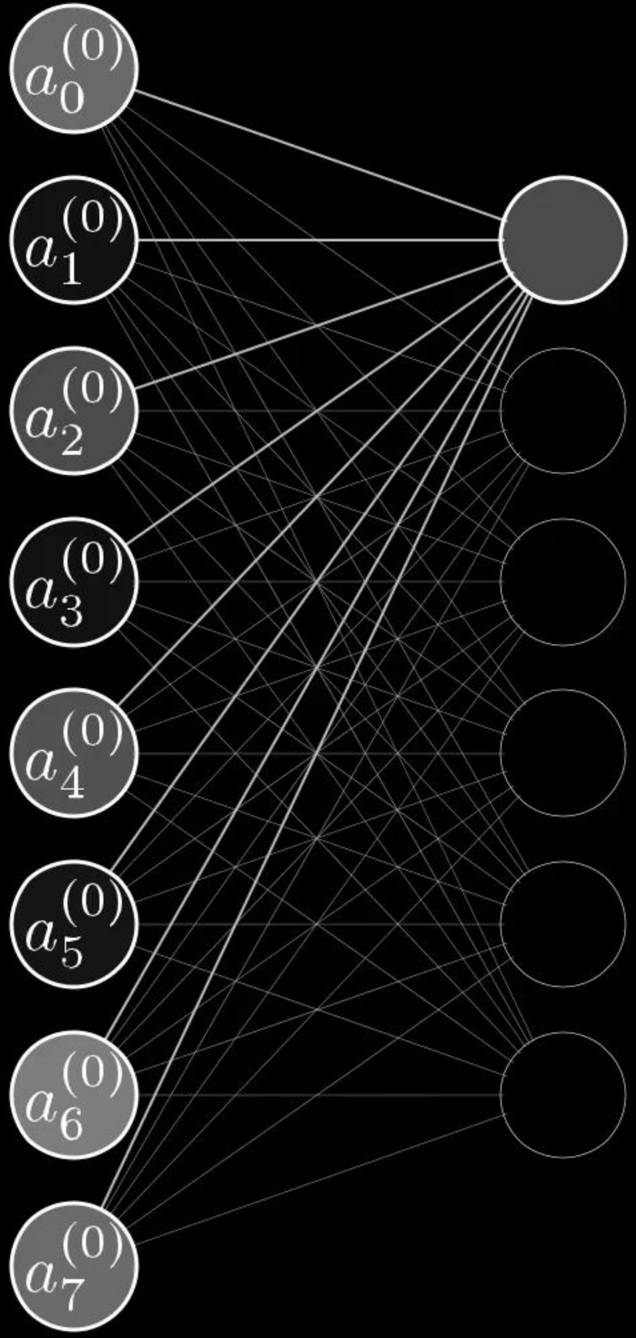


$$a_0^{(1)} = \overset{\text{Sigmoid}}{\sigma}\left(w_{0,0}\; a_0^{(0)} + w_{0,1}\; a_1^{(0)} + \cdots + w_{0,n}\; a_n^{(0)} + \underset{\text{Bias}}{b_0} \right)$$



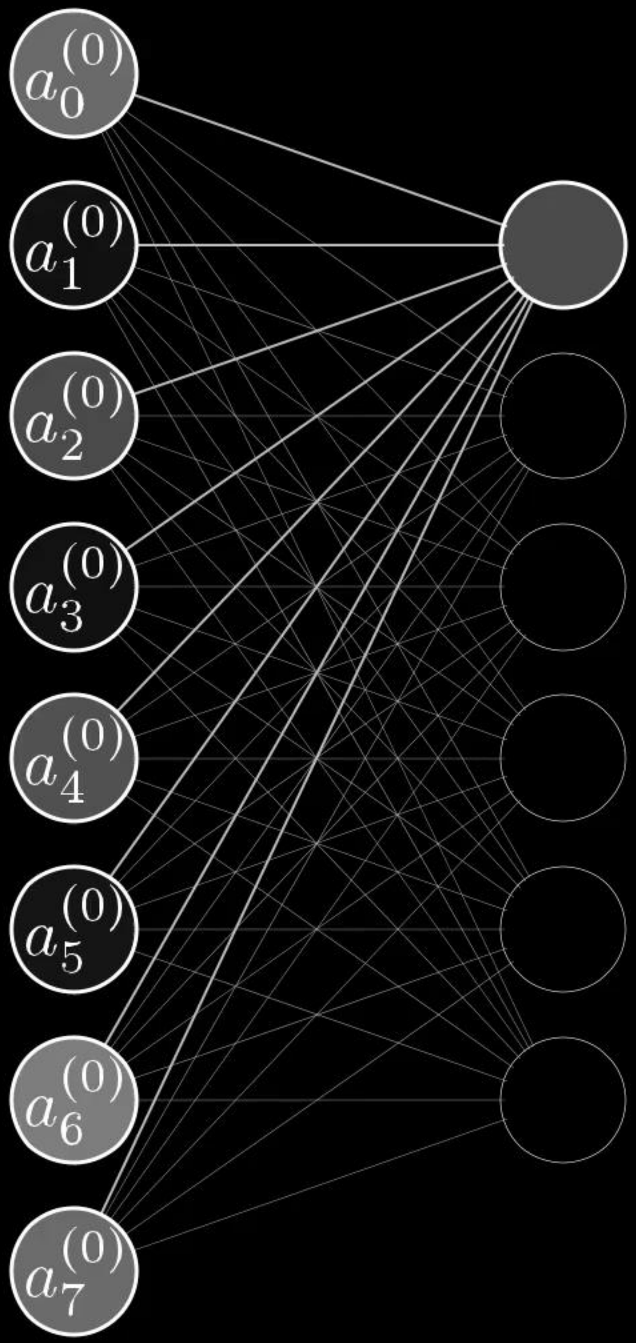
$$a_0^{(1)} = \overset{\substack{\text{Sigmoid} \\ \downarrow}}{\sigma}\left(w_{0,0}\, a_0^{(0)} + w_{0,1}\, a_1^{(0)} + \cdots + w_{0,n}\, a_n^{(0)} + \underset{\substack{\uparrow \\ \text{Bias}}}{b_0} \right)$$

$$\begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix}$$



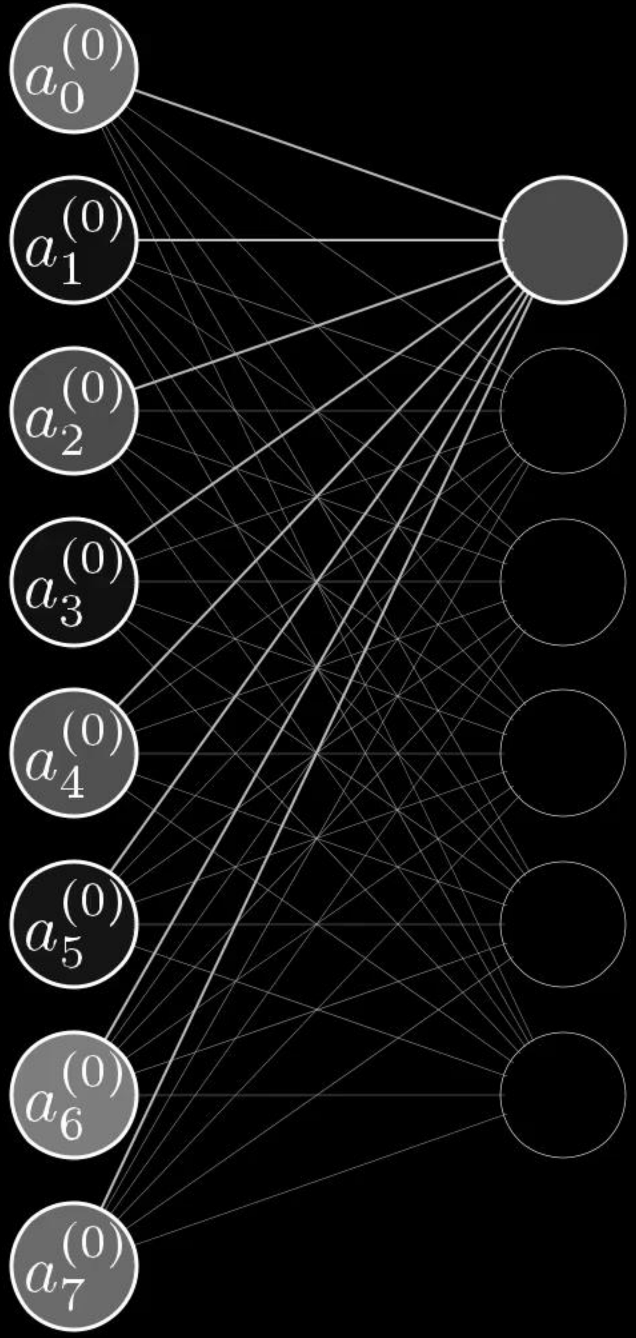
$$a_0^{(1)} = \overset{\text{Sigmoid}}{\overset{\downarrow}{\sigma}}\left(w_{0,0}\, a_0^{(0)} + w_{0,1}\, a_1^{(0)} + \cdots + w_{0,n}\, a_n^{(0)} + \underset{\text{Bias}}{\underset{\uparrow}{b_0}} \right)$$

$$\begin{bmatrix} \boxed{w_{0,0} \quad w_{0,1} \quad \cdots \quad w_{0,n}} \\ w_{1,0} \quad w_{1,1} \quad \cdots \quad w_{1,n} \\ \vdots \qquad \vdots \qquad \ddots \qquad \vdots \\ w_{k,0} \quad w_{k,1} \quad \cdots \quad w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix}$$



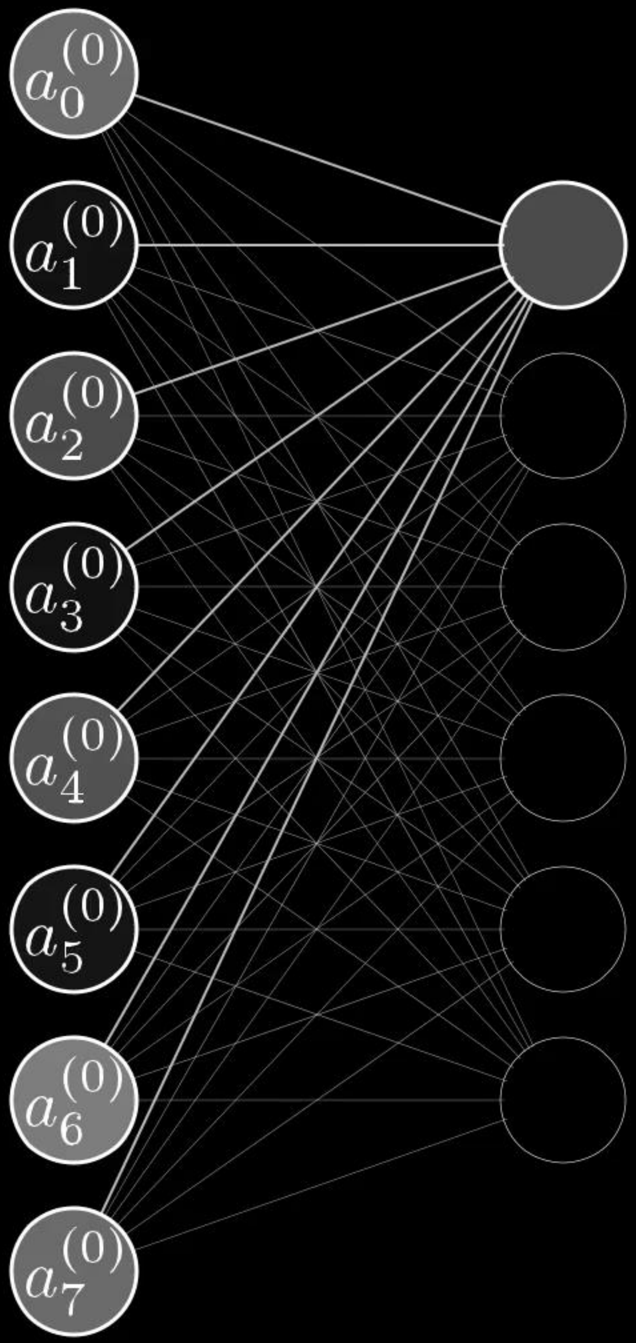
$$a_0^{(1)} = \overset{\substack{\text{Sigmoid}\\ \downarrow}}{\sigma}\left(w_{0,0}\ a_0^{(0)} + w_{0,1}\ a_1^{(0)} + \cdots + w_{0,n}\ a_n^{(0)} \underset{\substack{\uparrow\\ \text{Bias}}}{\boxed{+ b_0}} \right)$$

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix}$$



$$a_0^{(1)} = \overset{\substack{\text{Sigmoid} \\ \downarrow}}{\sigma}\left(w_{0,0}\; a_0^{(0)} + w_{0,1}\; a_1^{(0)} + \cdots + w_{0,n}\; a_n^{(0)} + \underset{\substack{\uparrow \\ \text{Bias}}}{b_0} \right)$$

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}$$

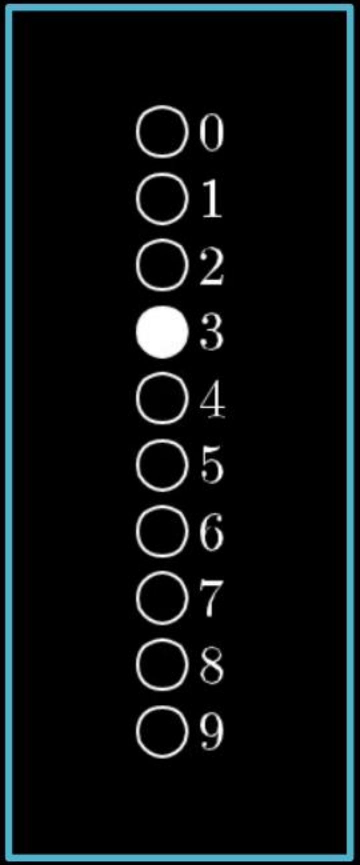


$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$

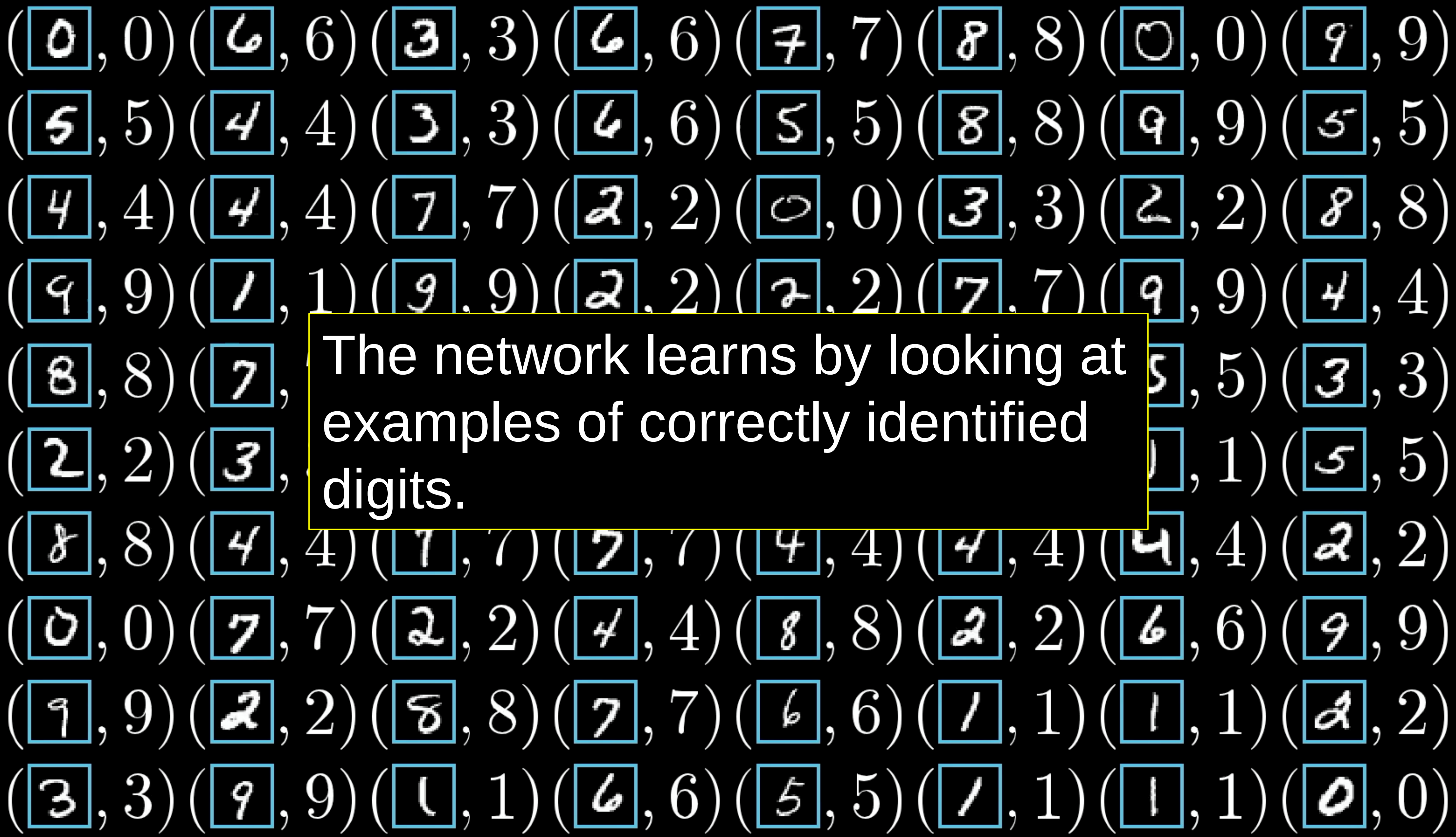
Neural network function

784 inputs

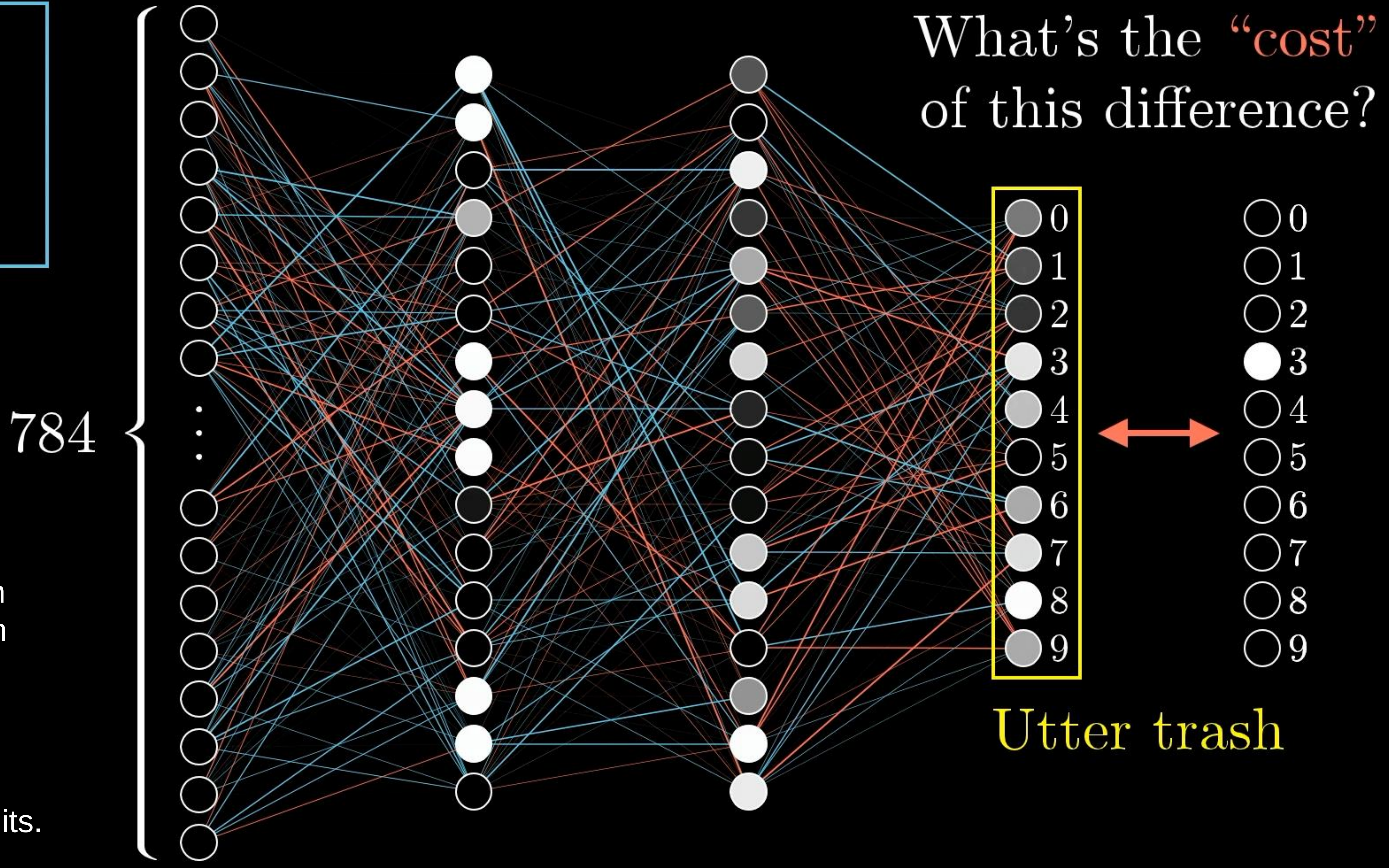
the entire network is just a **function**! It takes in 784 numbers as its input, and spits out 10 numbers as its output. It’s an absurdly complicated function, because it takes over 13,000 parameters (weights and biases), and it involves iterating many matrix-vector products and sigmoid squishificaitons together. But it’s just a function nonetheless.



10 outputs

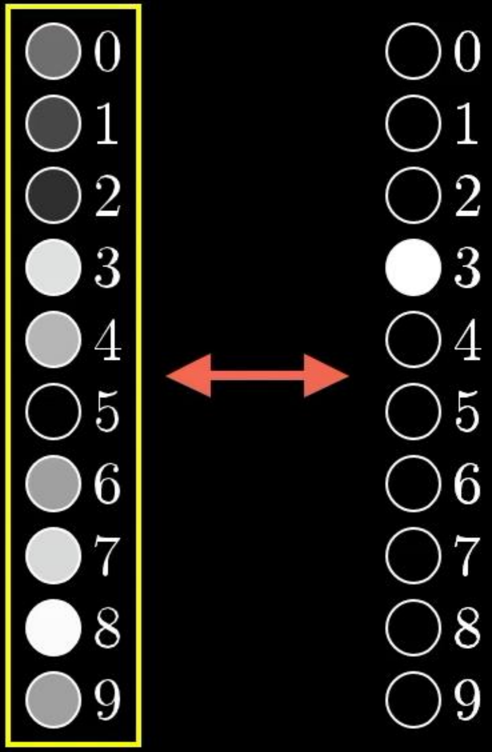


The network learns by looking at examples of correctly identified digits.



Initialized with totally random weights and biases, the network is terrible at identifying digits.

What's the "cost" of this difference?

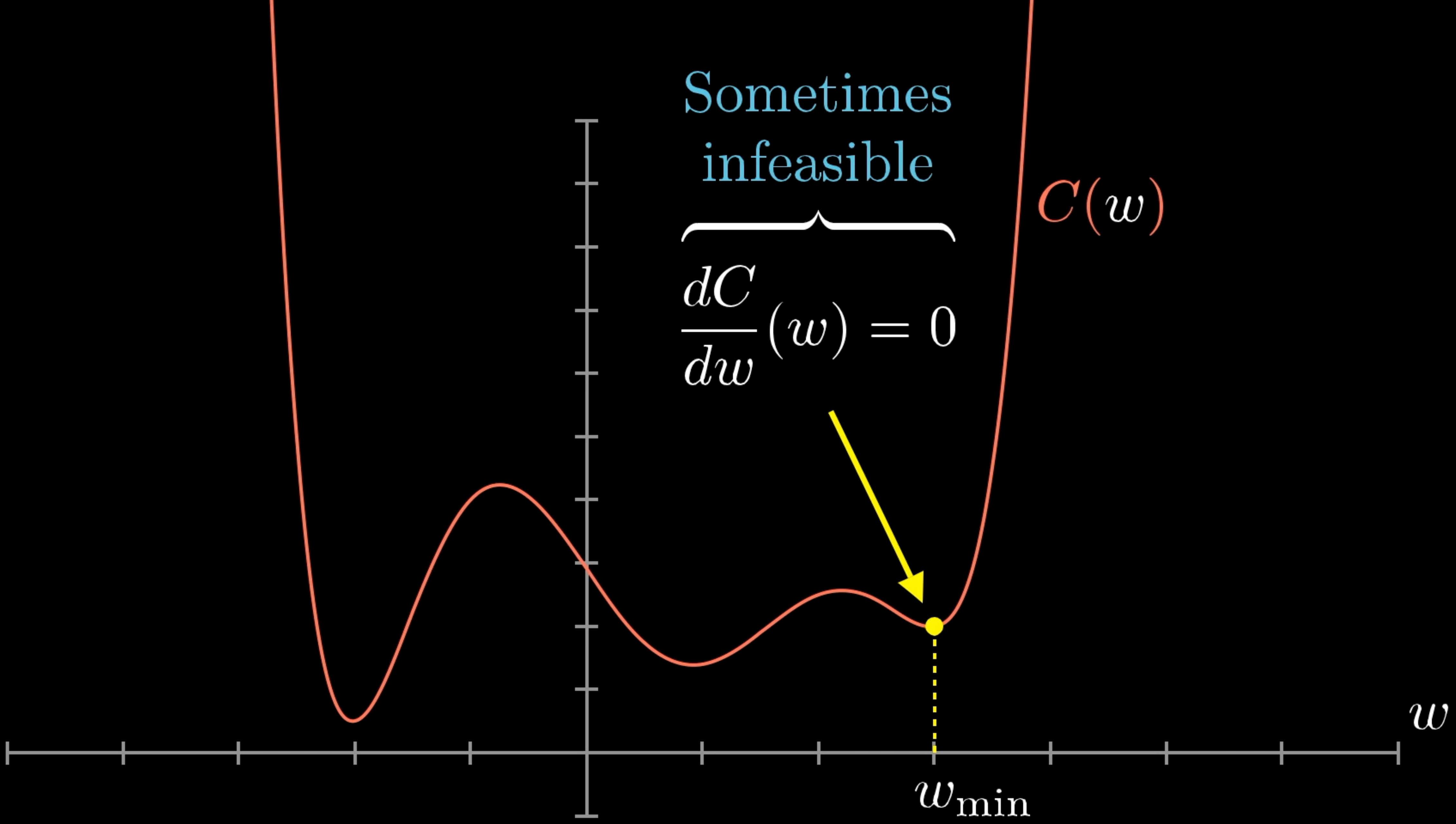


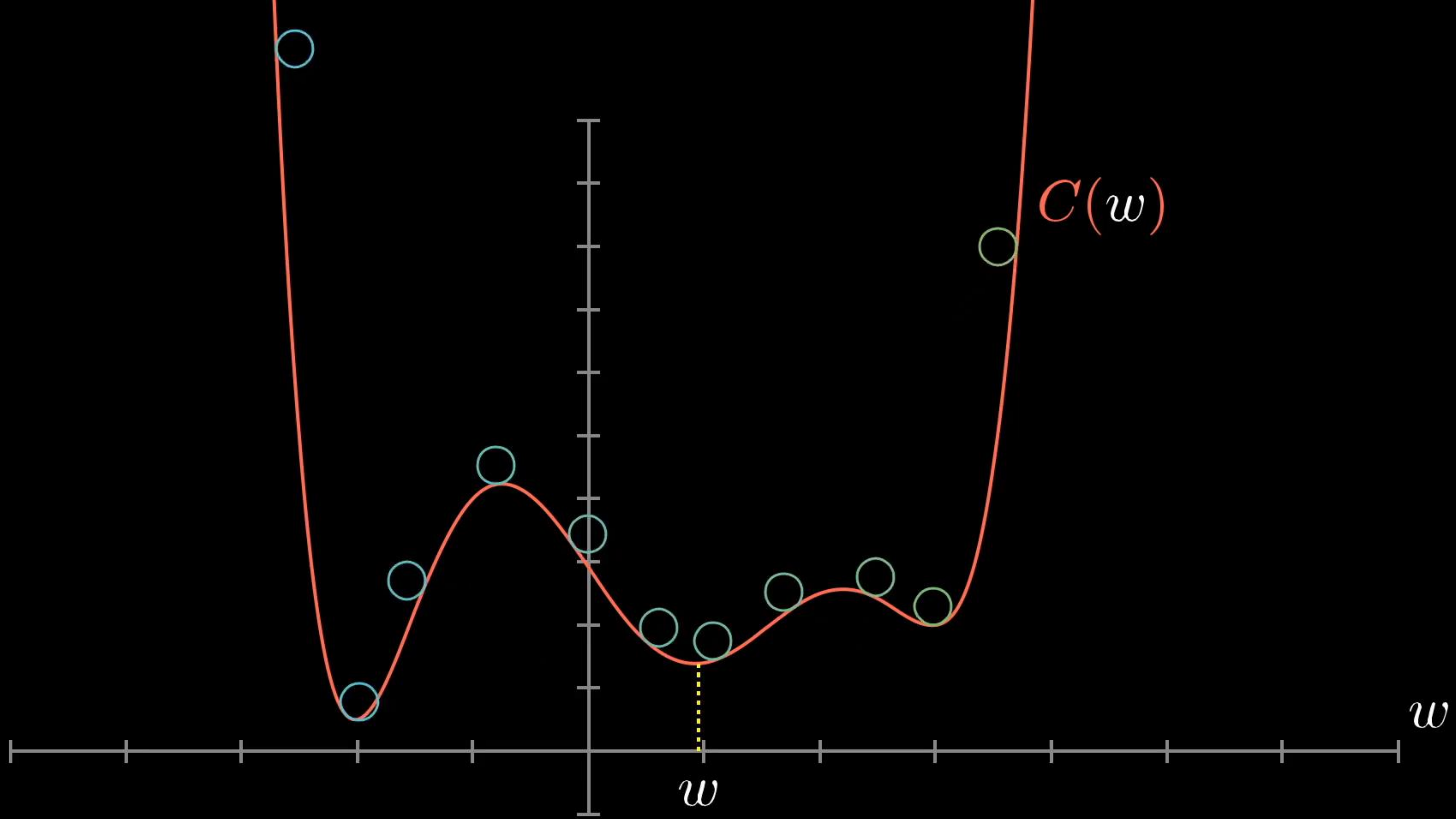
Utter trash

Cost of 

$$\begin{cases} (0.43 - 0.00)^2 + \\ (0.28 - 0.00)^2 + \\ (0.19 - 0.00)^2 + \\ (0.88 - 1.00)^2 + \\ (0.72 - 0.00)^2 + \\ (0.01 - 0.00)^2 + \\ (0.64 - 0.00)^2 + \\ (0.86 - 0.00)^2 + \\ (0.99 - 0.00)^2 + \\ (0.63 - 0.00)^2 \end{cases}$$

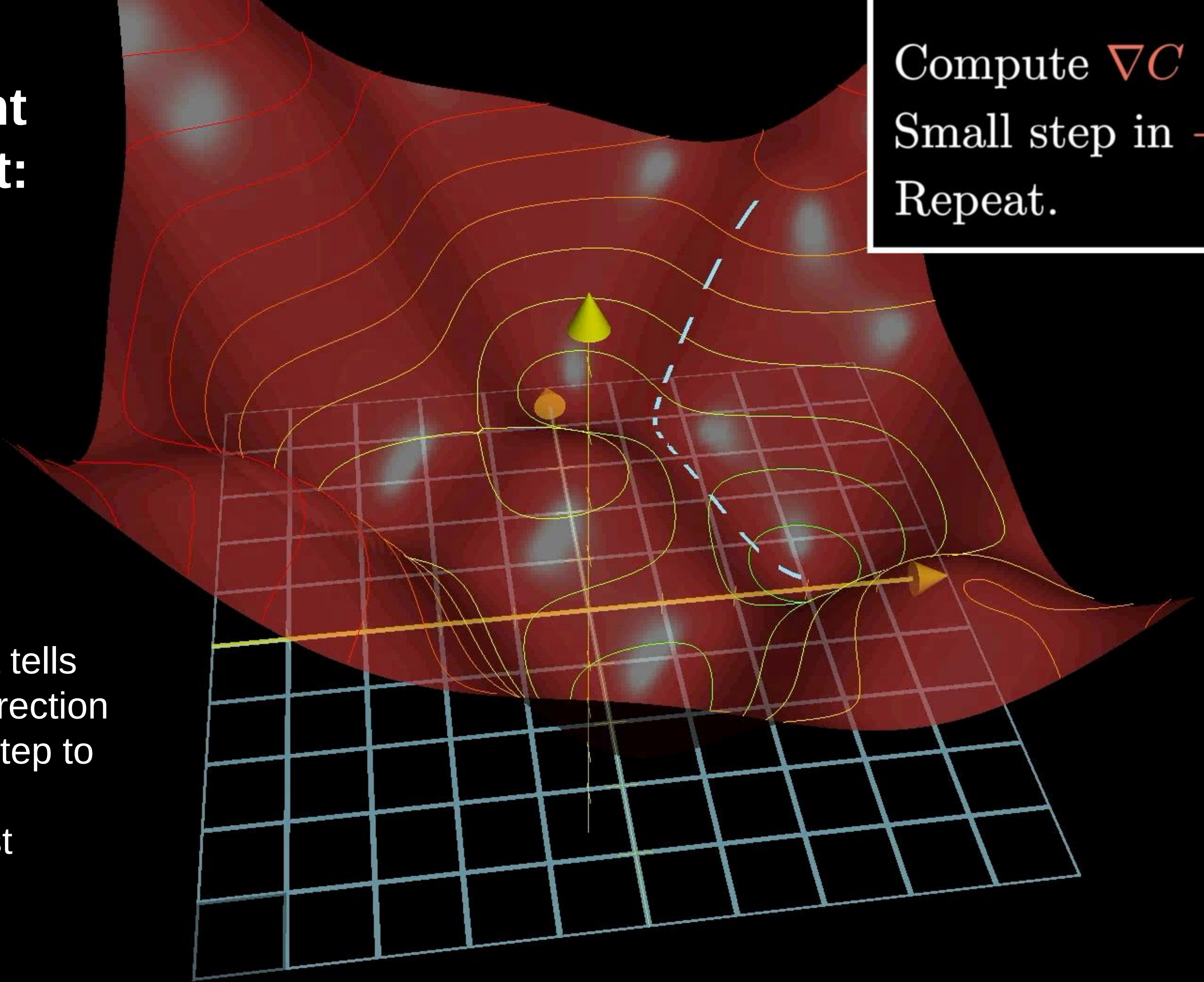
The "cost" is calculated by adding up the squares of the differences between what we got and what we want.


Sometimes infeasible
$\frac{dC}{dw}(w) = 0$
$C(w)$
w
w_{min}



$C(w)$
w
w

Gradient descent:



Compute ∇C

Small step in $-\nabla C$ direction

Repeat.

The gradient tells you which direction you should step to increase the function most quickly.

Each step: $-\eta\nabla C$; η=learning rate;
Take shorter steps closer to the minima

13,002 weights and biases

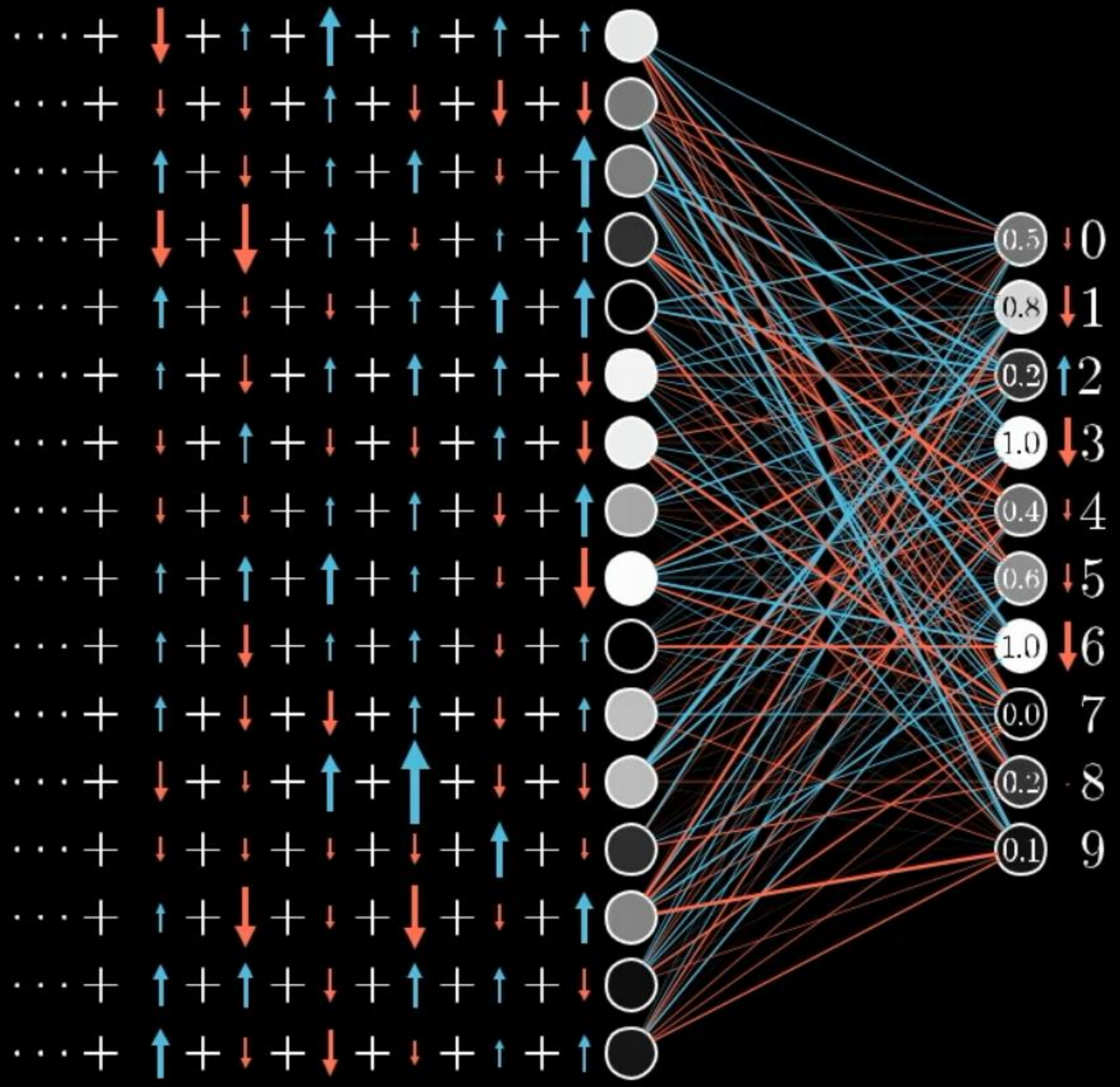
How to nudge all weights and biases

$$\vec{\mathbf{W}} = \begin{bmatrix} 2.25 \\ -1.57 \\ 1.98 \\ \vdots \\ -1.16 \\ 3.82 \\ 1.21 \end{bmatrix} \qquad -\nabla C(\vec{\mathbf{W}}) = \begin{bmatrix} 0.18 \\ 0.45 \\ -0.51 \\ \vdots \\ 0.40 \\ -0.32 \\ 0.82 \end{bmatrix}$$

$$0.2 = \sigma(w_0 a_0 + w_1 a_1 + \cdots + w_{n-1} a_{n-1} + b)$$

There are three avenues that can team up together to increase this activation:

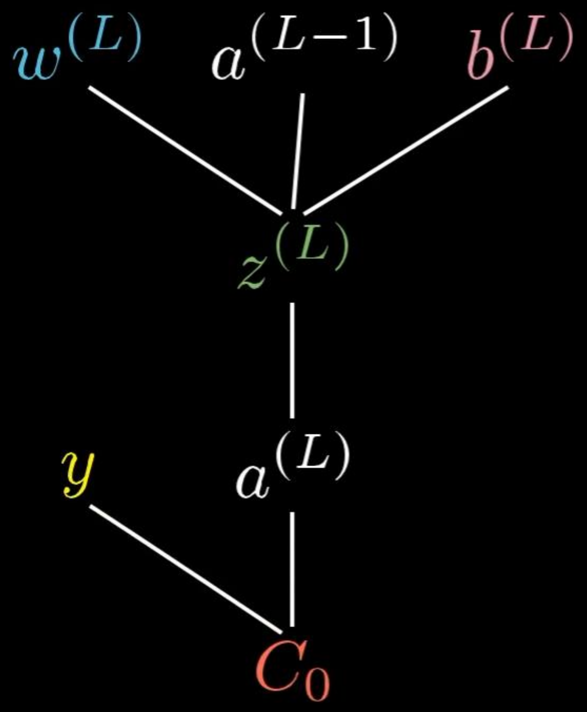
1. Increase the bias
2. Increase the weights
3. Change the activations from the previous layer

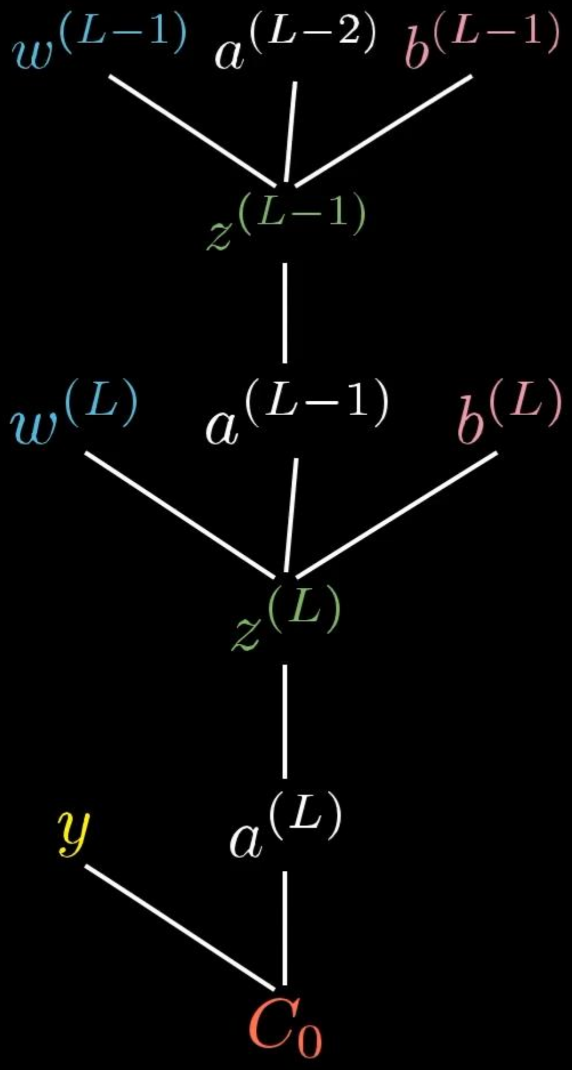


It’s impossible to perfectly satisfy all these competing desires for activations in the second layer.

The best we can do is add up all the desired nudges to find the overall desired change.

Calculating the Gradient with Backpropagation:





$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix}$$

$$\nabla C \leftarrow \begin{cases} \dfrac{\partial C}{\partial w_{jk}^{(l)}} = a_k^{(l-1)} \sigma'\left(z_j^{(l)}\right) \boxed{\dfrac{\partial C}{\partial a_j^{(l)}}} \\ \\ \boxed{\begin{array}{c} \displaystyle\sum_{j=0}^{n_{l+1}-1} w_{jk}^{(l+1)} \sigma'\left(z_j^{(l+1)}\right) \dfrac{\partial C}{\partial a_j^{(l+1)}} \\ \\ \text{or} \\ \\ 2\left(a_j^{(L)} - y_j\right) \end{array}} \end{cases}$$