Neural Networks

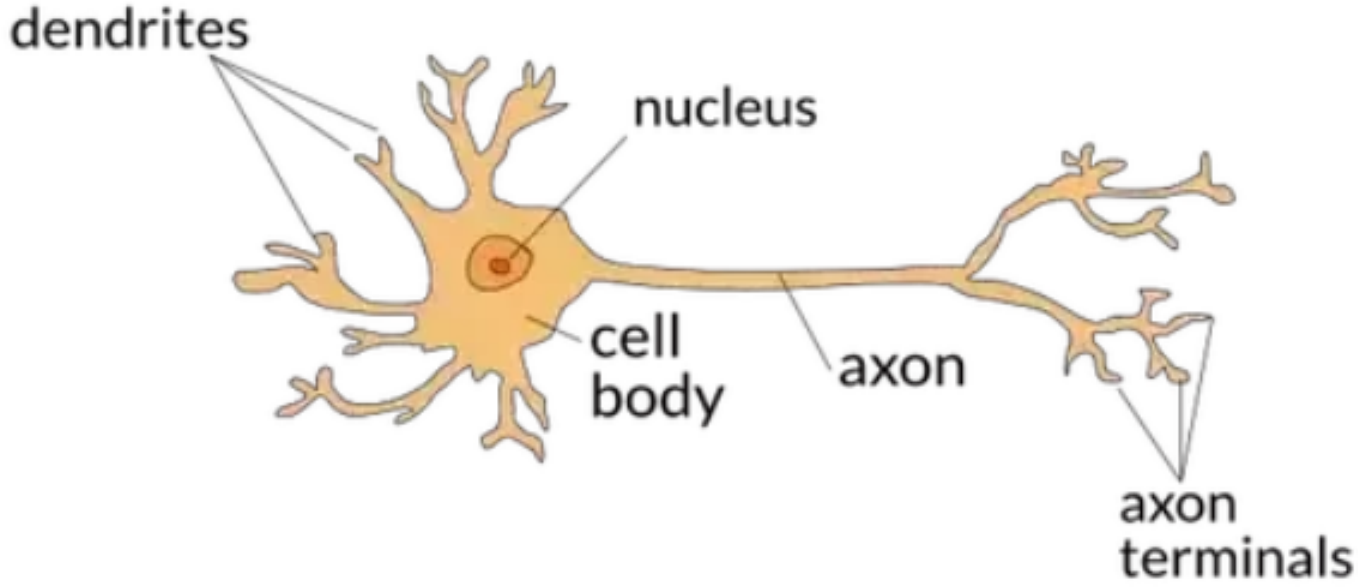
Neural Networks

- Neurons & Perceptrons
- Perceptron Neural Networks
  - Layers
  - Activation Functions
  - Softmax
  - Cost Functions
- Backpropagation
  - Stochastic Gradient Descent

Neural Networks, 3Blue1Brown

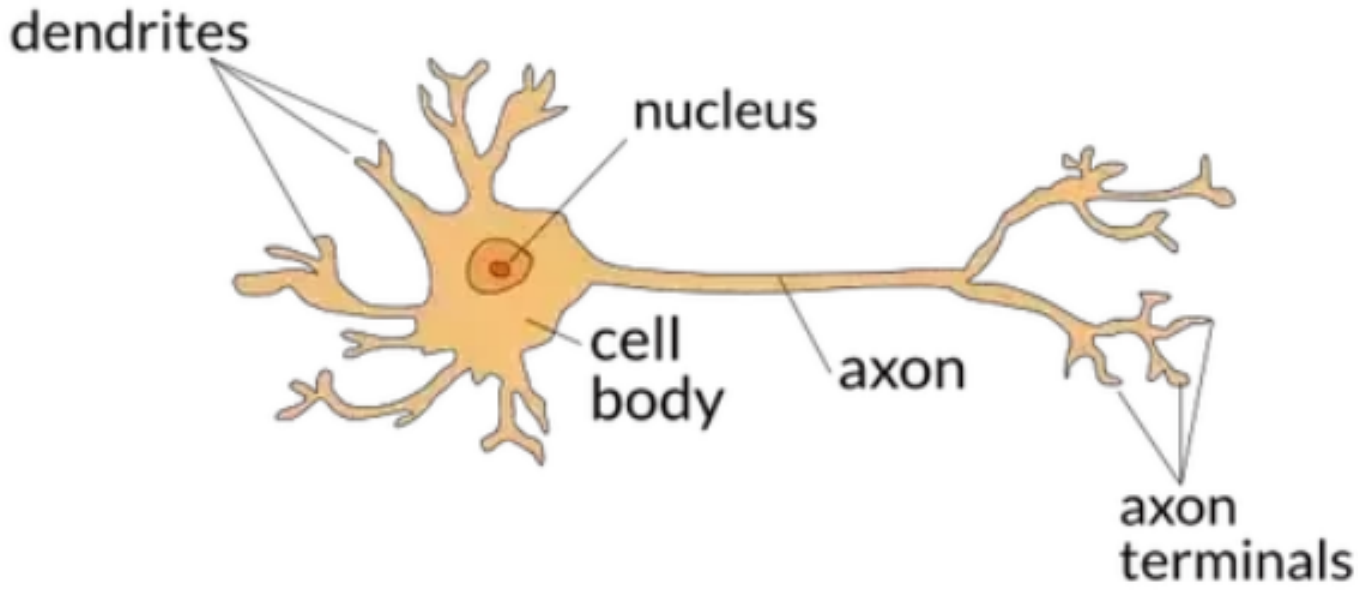
Neurons

- The neuron is the basic building block of a neural network.
- In a basic neural network, an artificial neuron mimics the biological neuron.



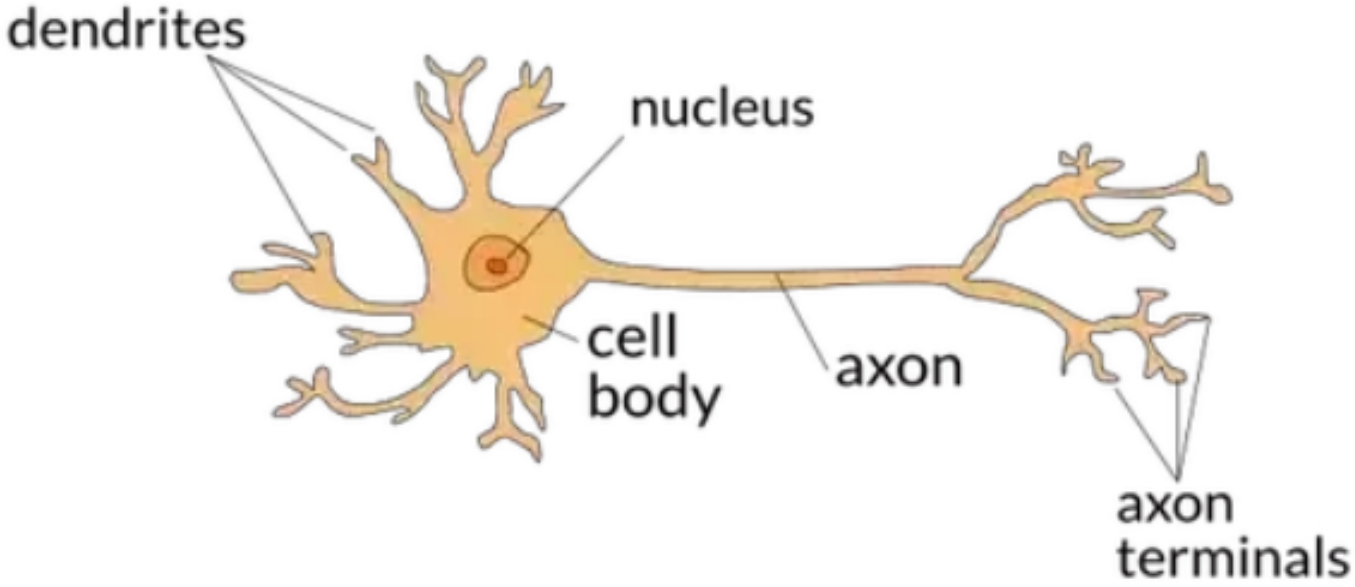
Neurons

- Signals can be received from dendrites
- Once enough signals are received, they are sent down the axon.
- This outgoing signal can then be used as another input for other neurons, repeating the process.



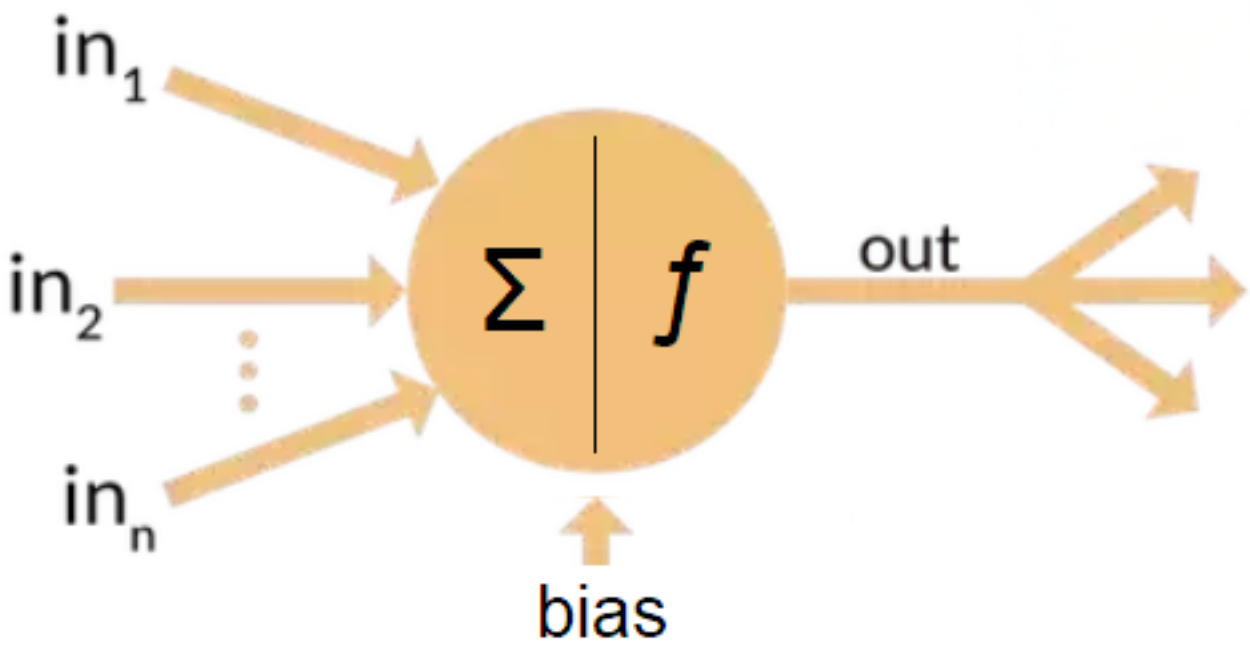
Neurons

- Some signals are more important than others and can trigger some neurons to fire easier.
- Connections can become stronger or weaker, new connections can appear while others can cease to exist.



Perceptrons

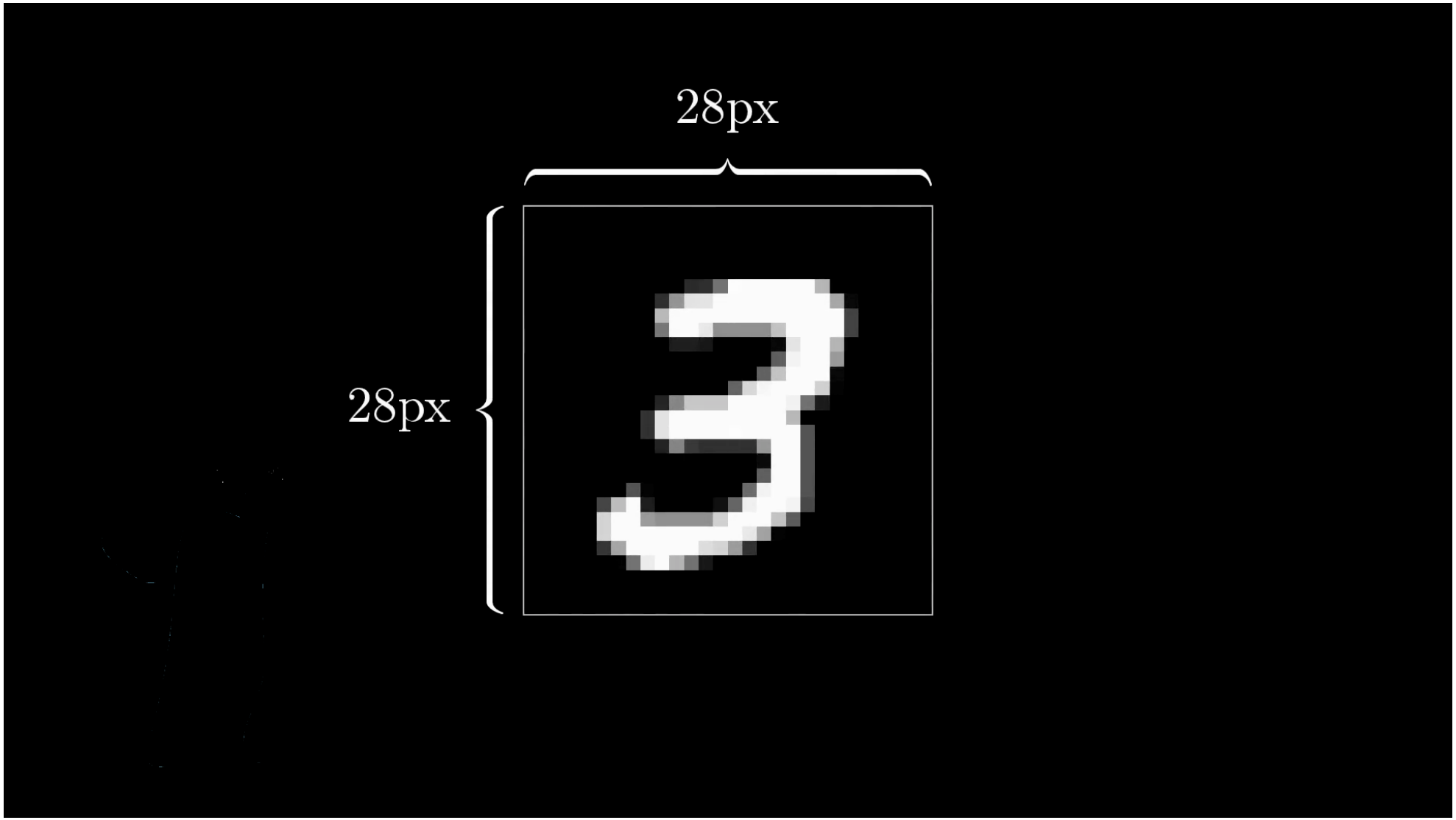
- We can mimic most of this process with the Perceptron
  - Receives a list of weighted input signals
  - Outputs some kind of signal if the sum of these weighted inputs reach a certain bias.



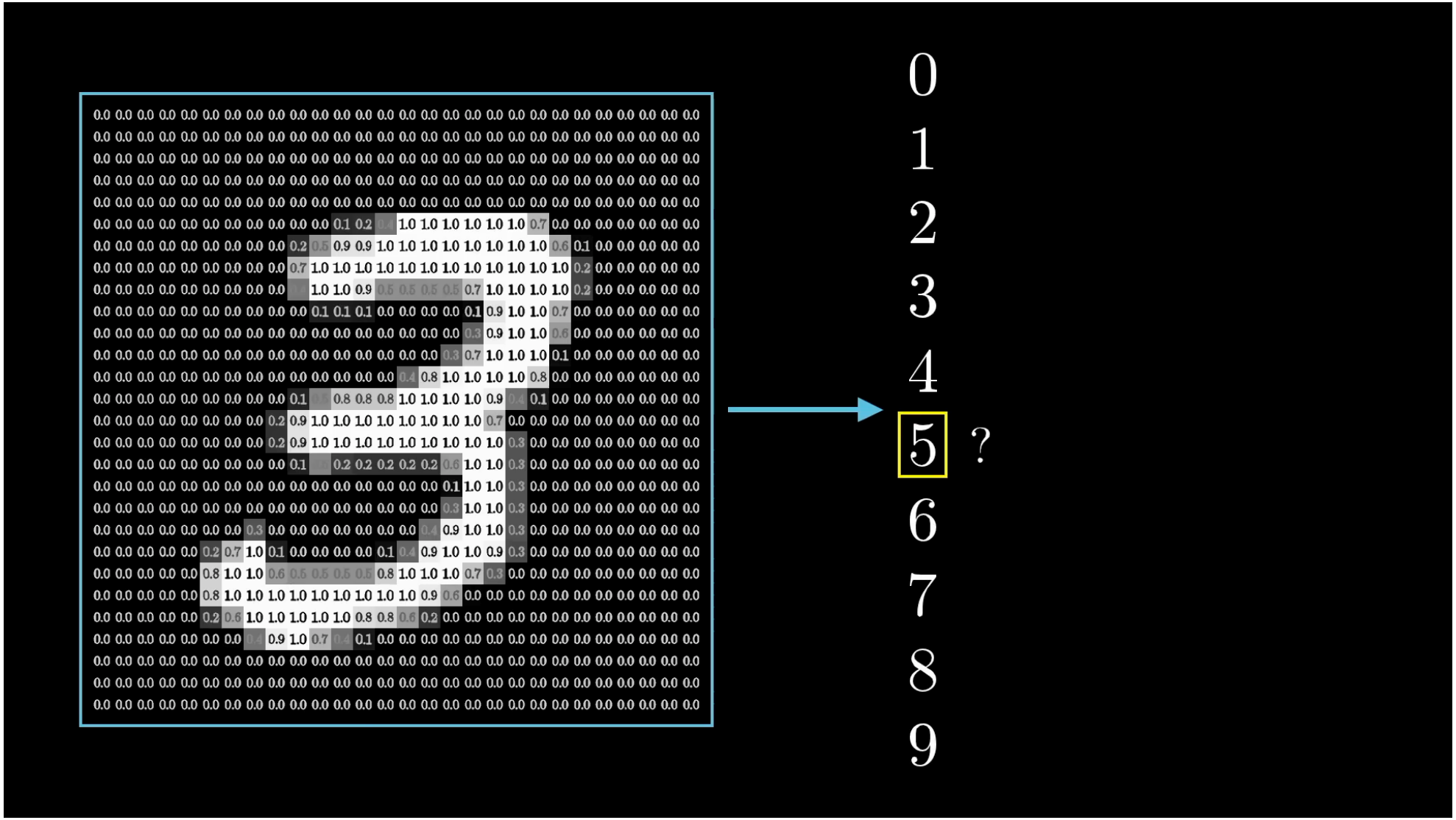
A Perceptron Neural Network

- The Challenge: train a model to recognize hand-written digits
- The input: Modified National Institute of Standard and Technology (MNIST) image database
  - 70,000 images

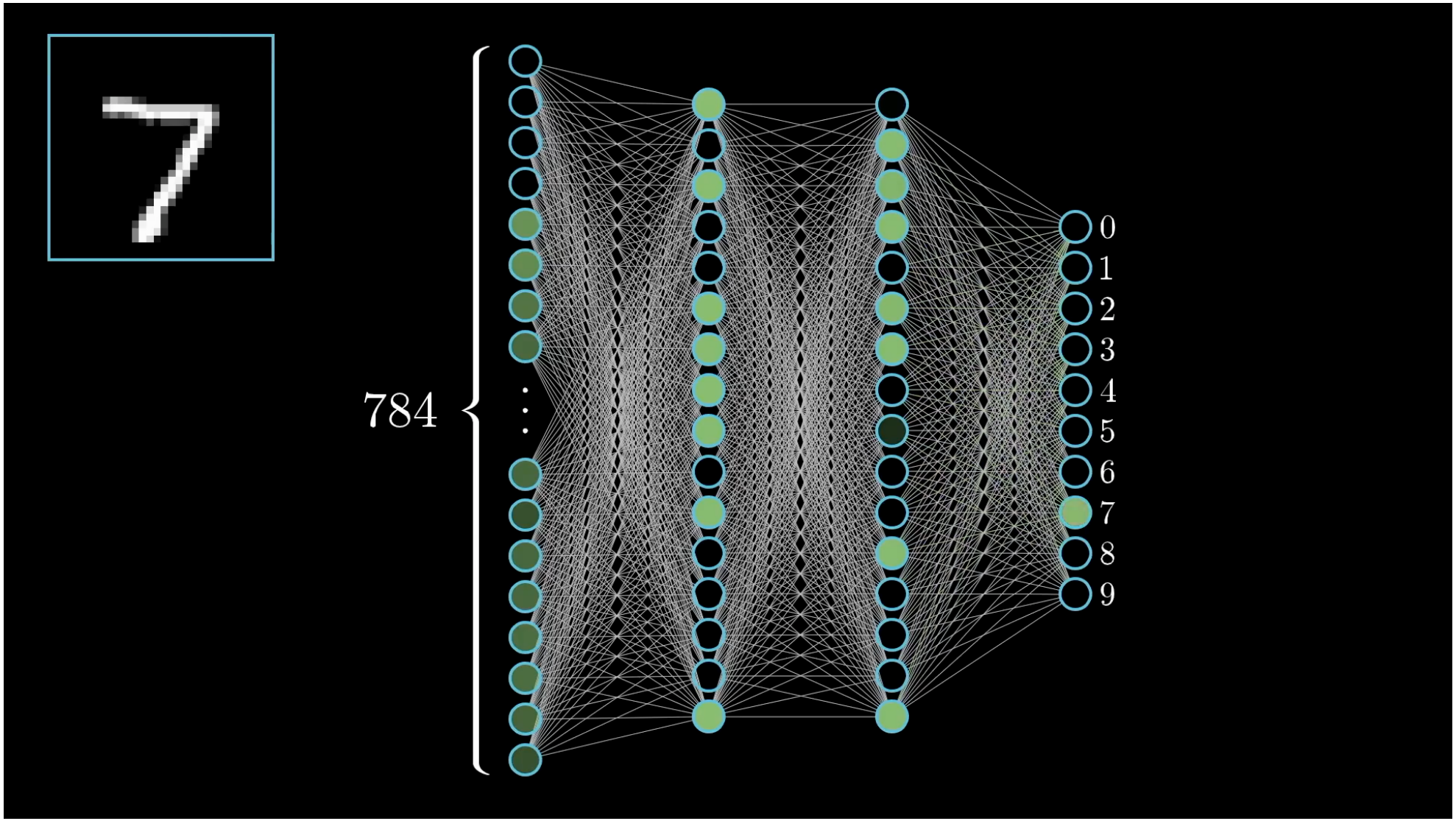
A Multi-Level Perceptron



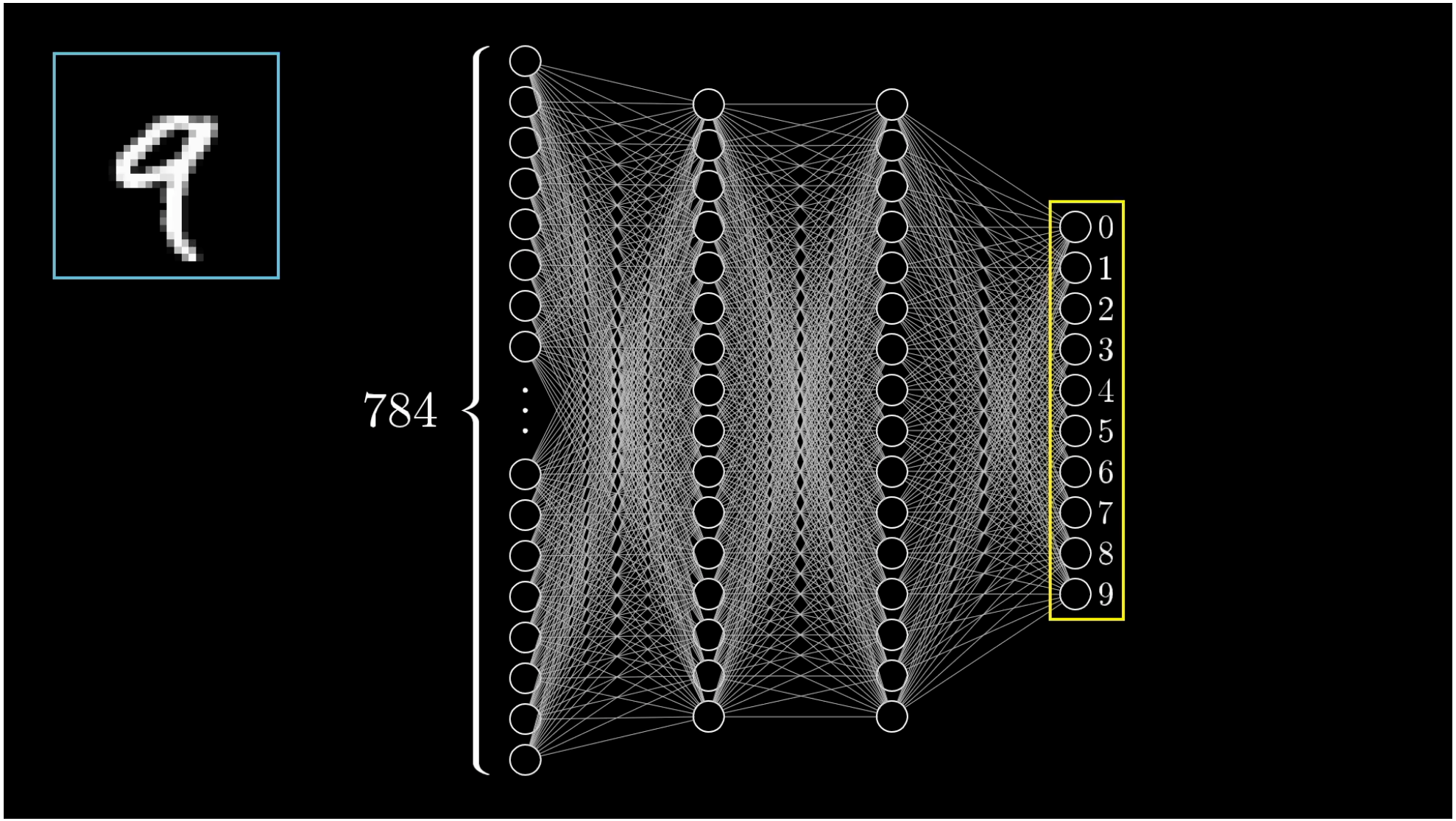
A Multi-Level Perceptron



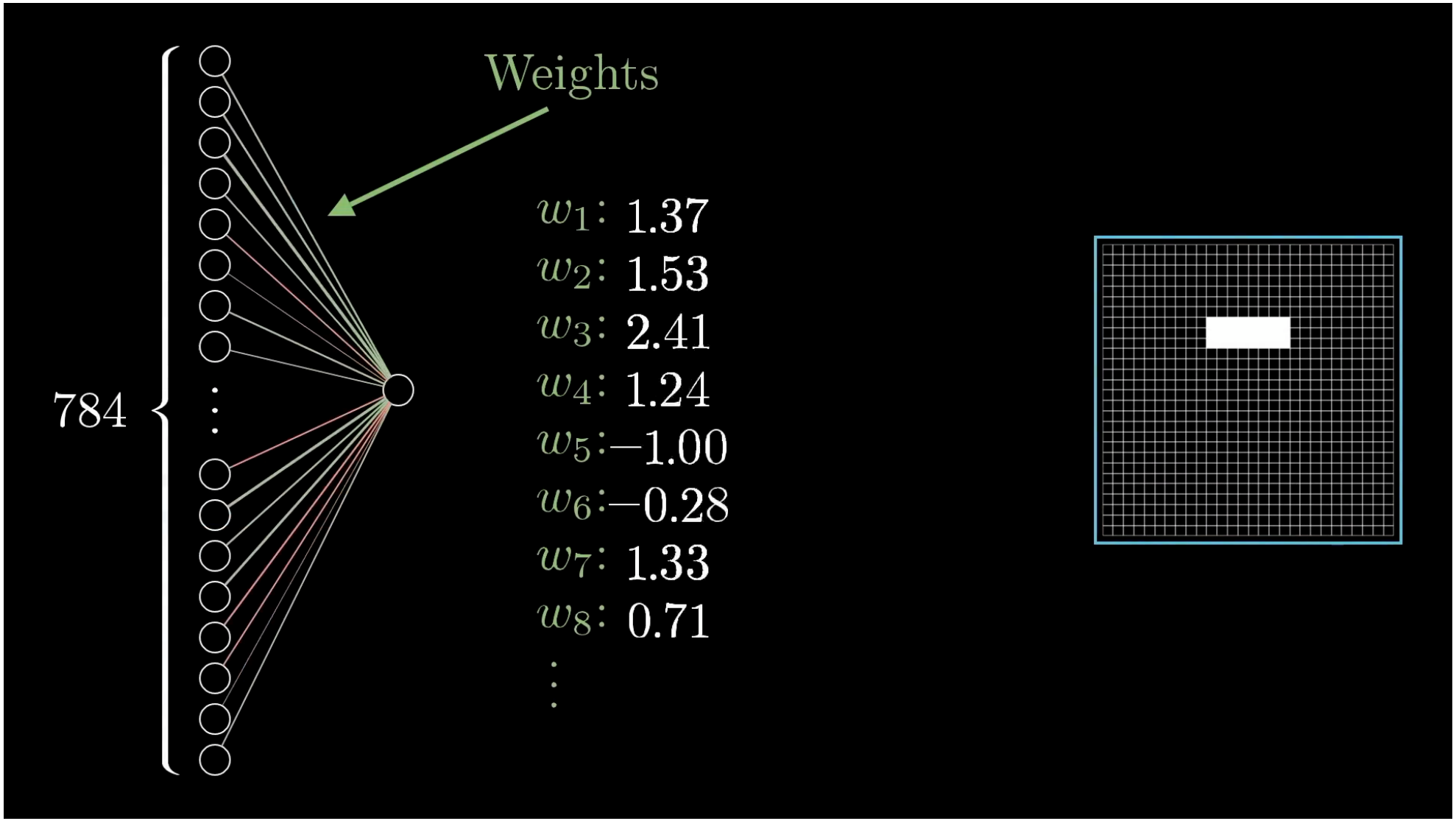
A Multi-Level Perceptron



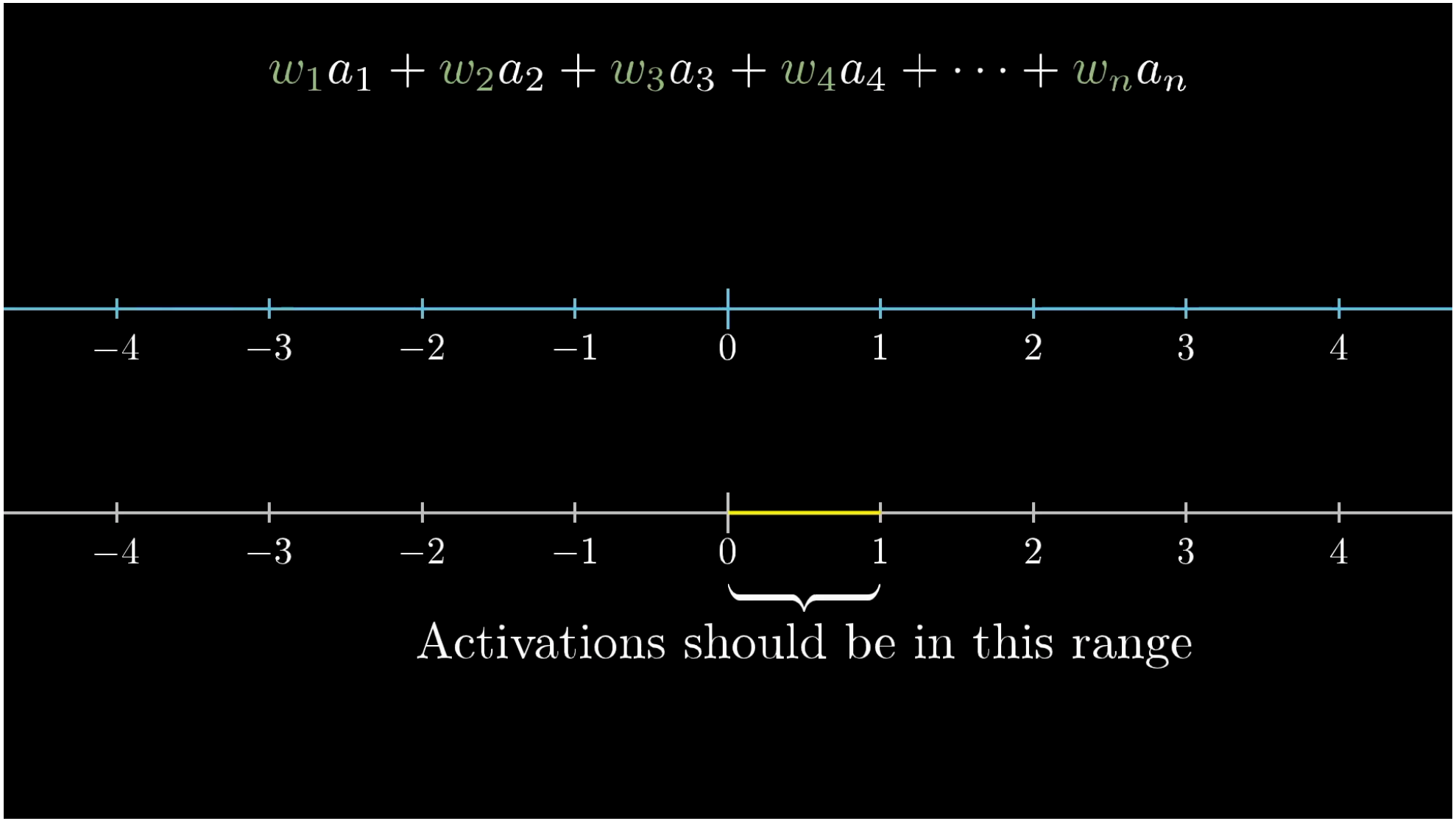
A Perceptron Neural Network



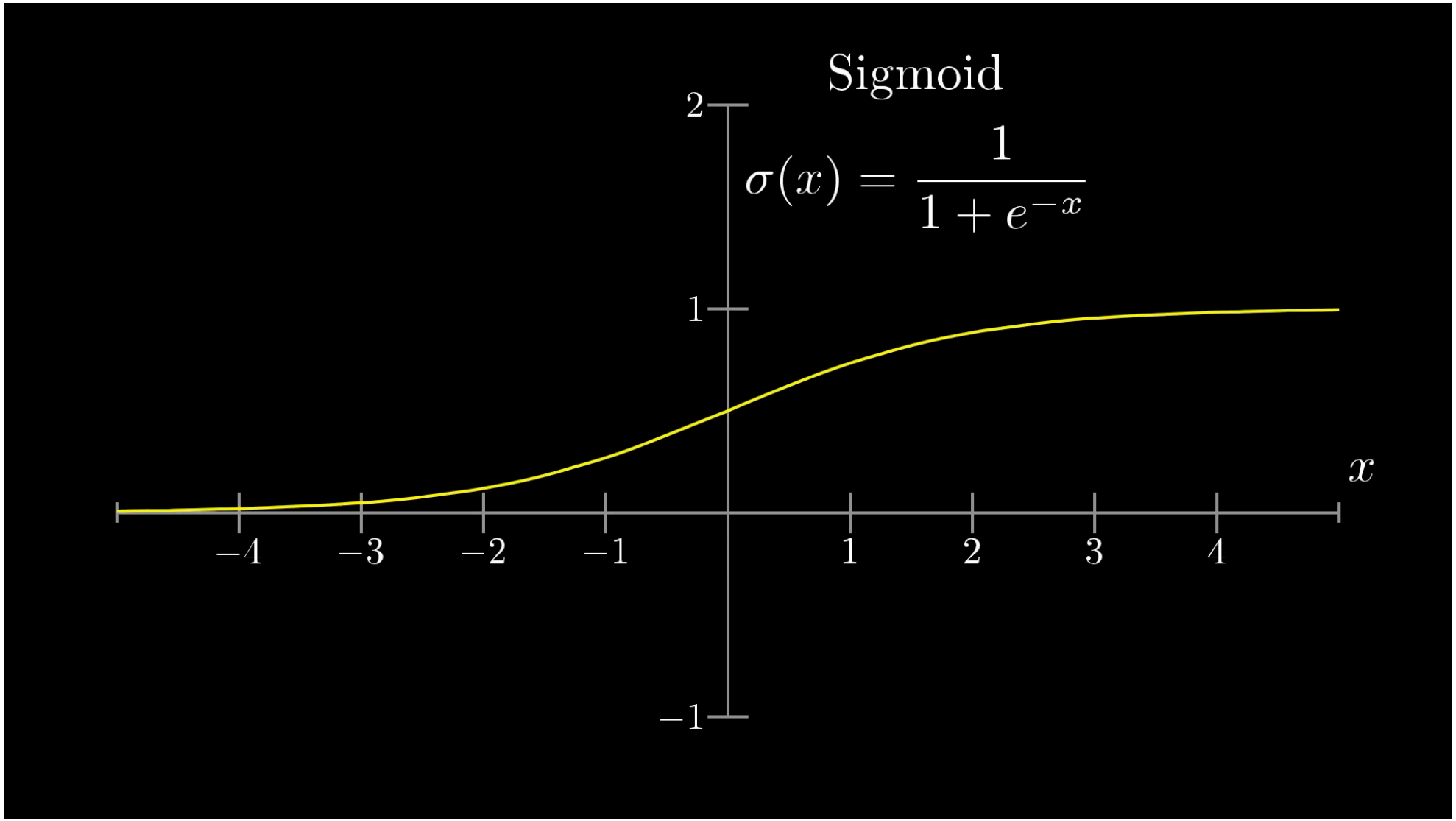
A Multi-Level Perceptron



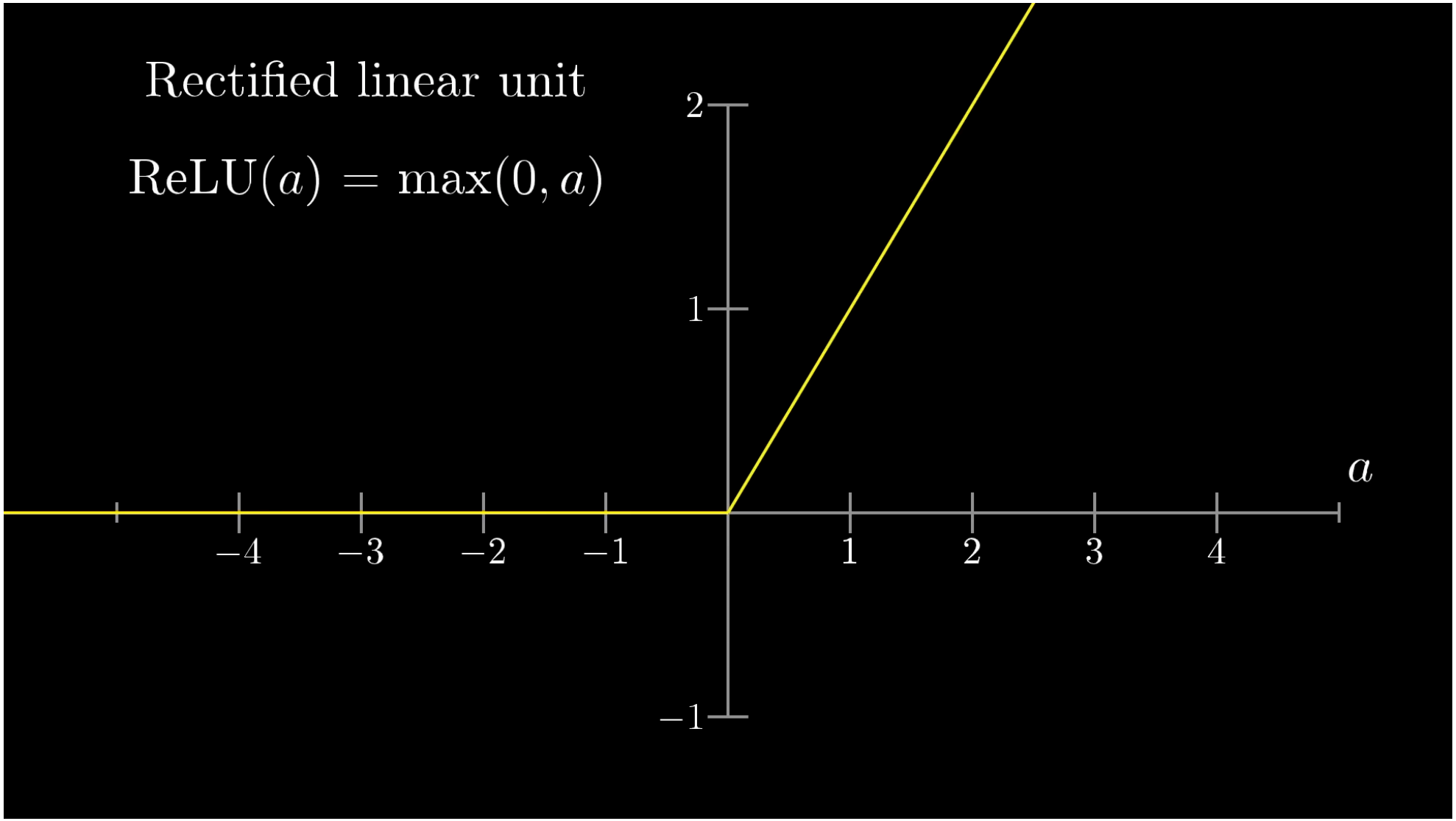
A Multi-Level Perceptron



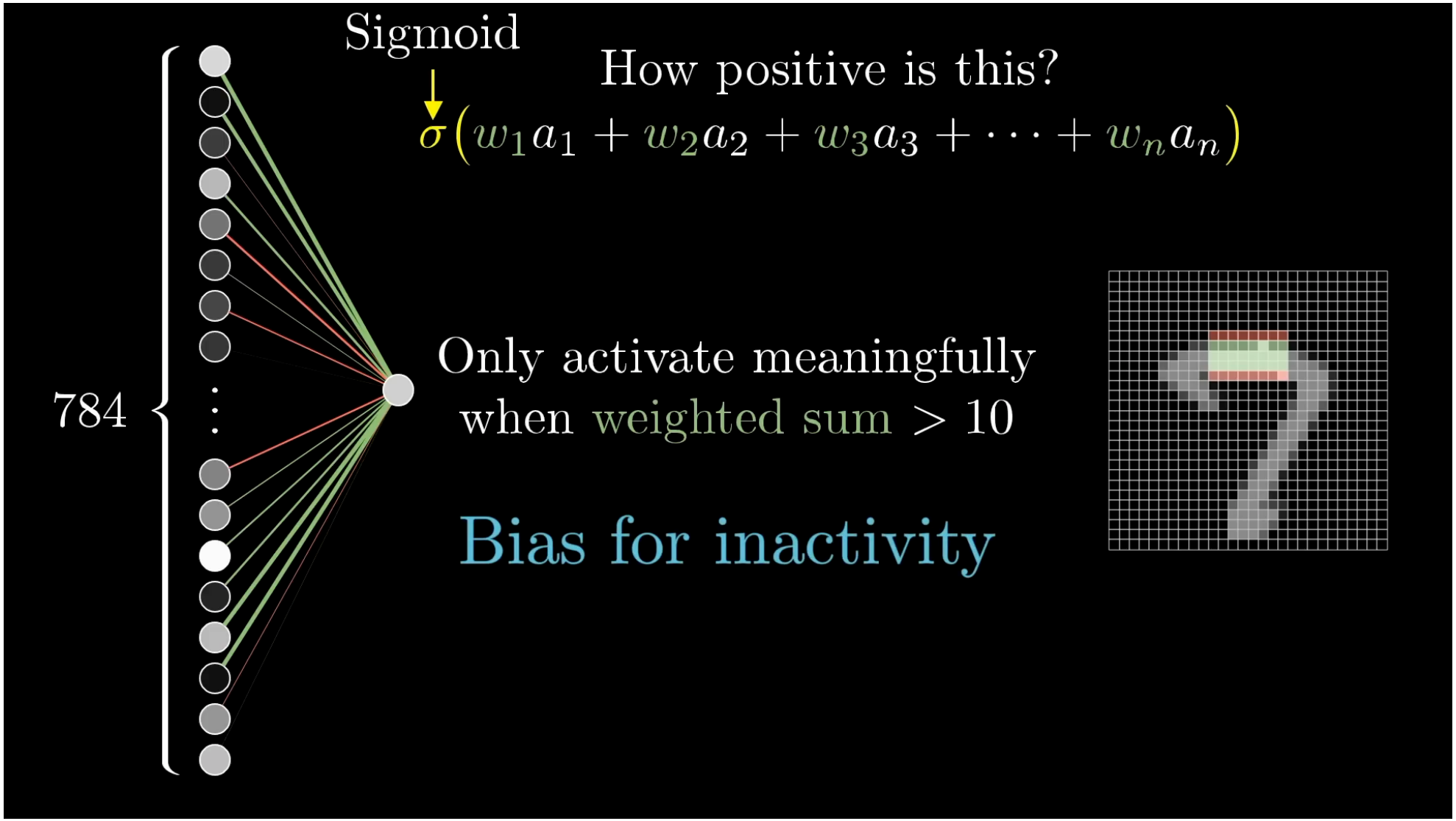
A Multi-Level Perceptron - Sigmoid Activation



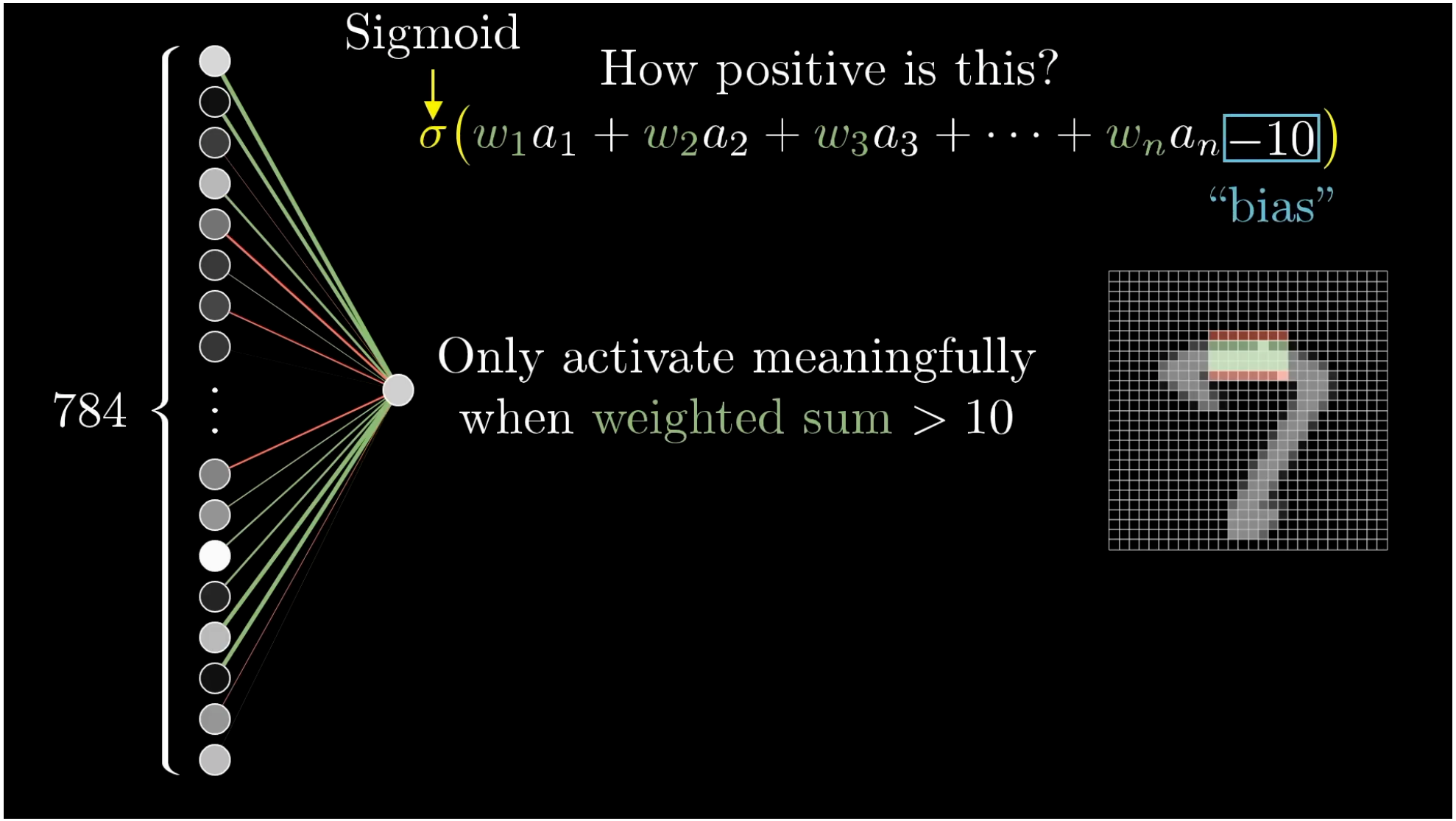
A Multi-Level Perceptron - ReLU Function



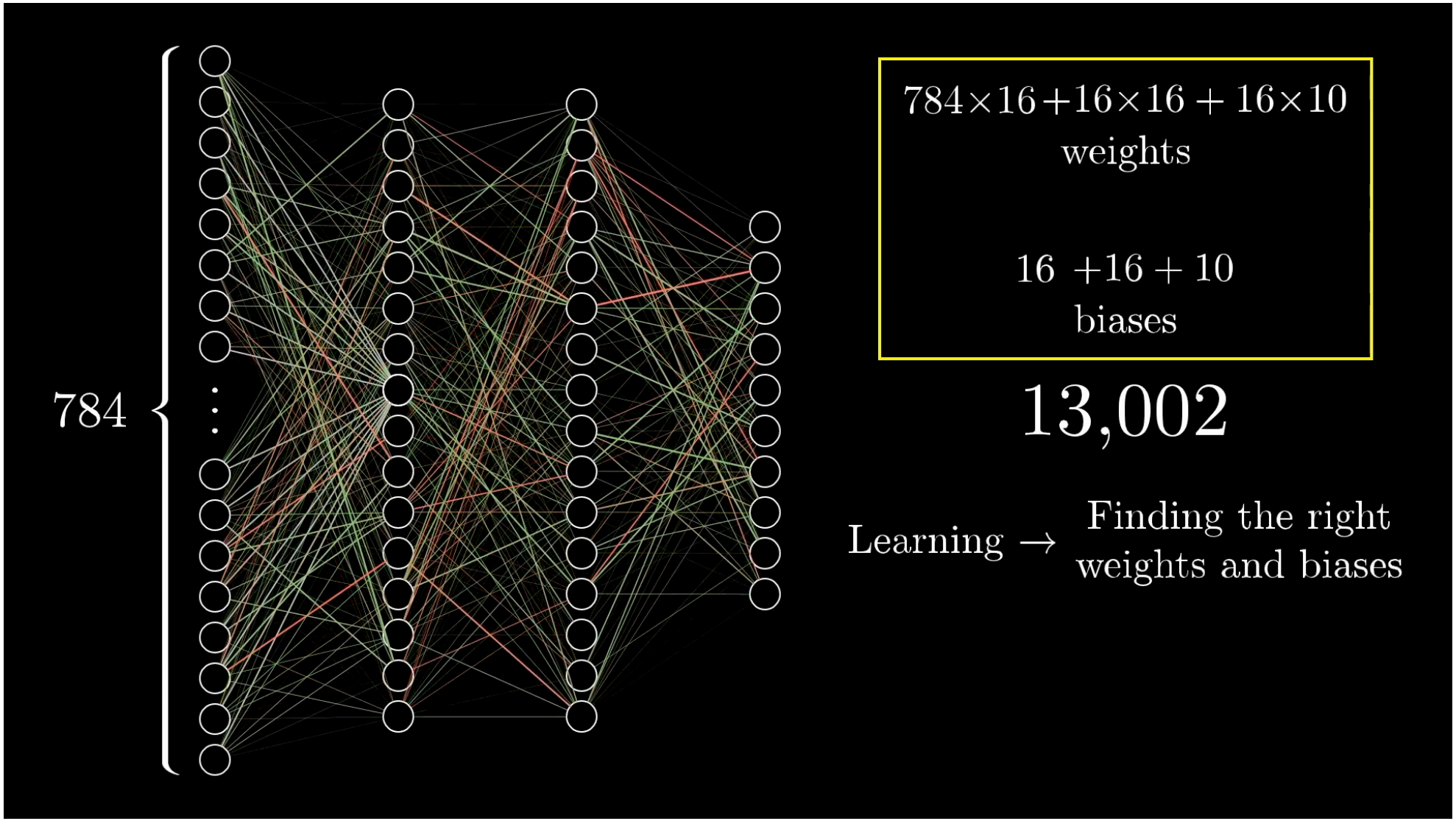
A Multi-Level Perceptron - Bias



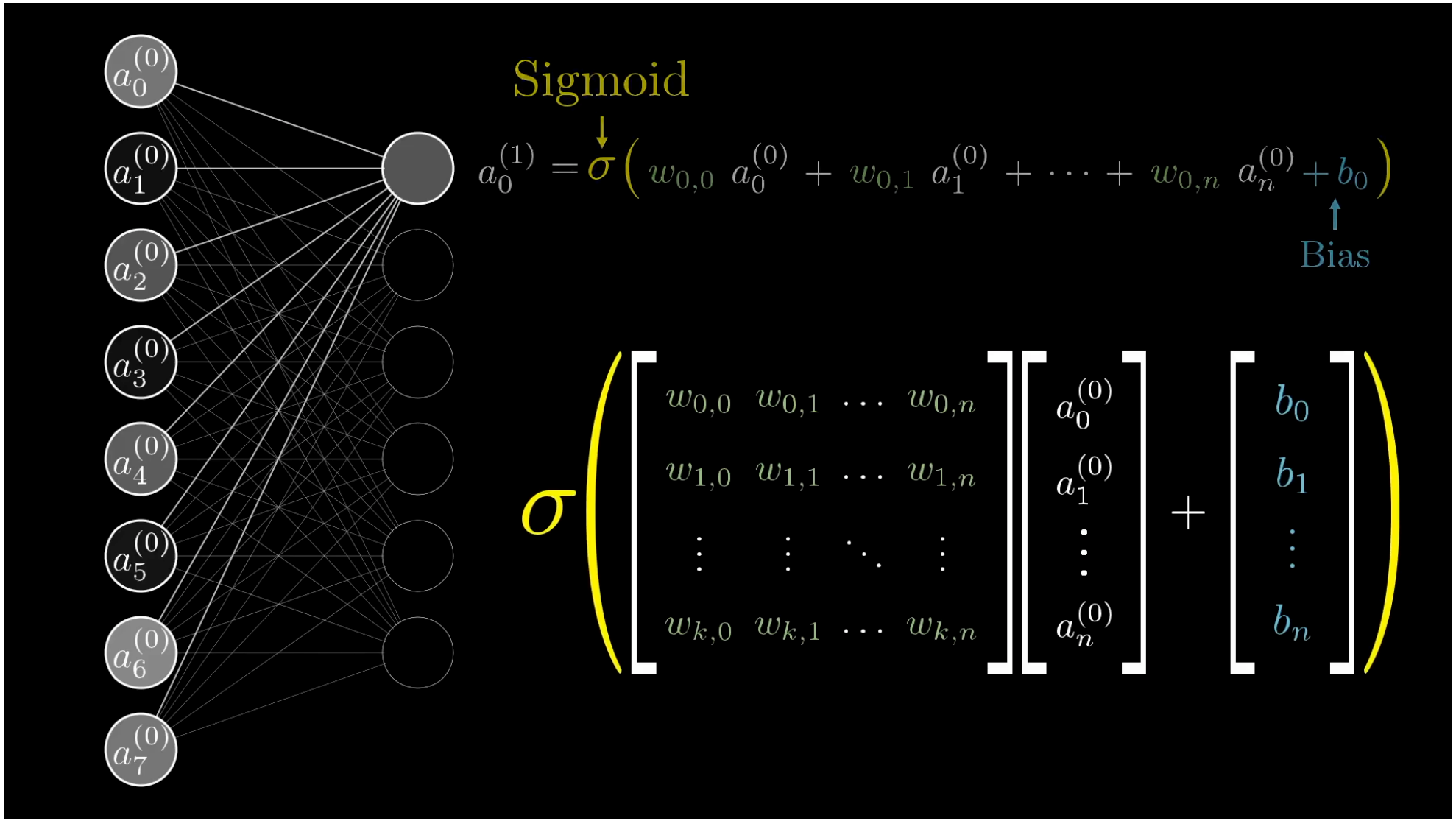
A Multi-Level Perceptron - Output Equation



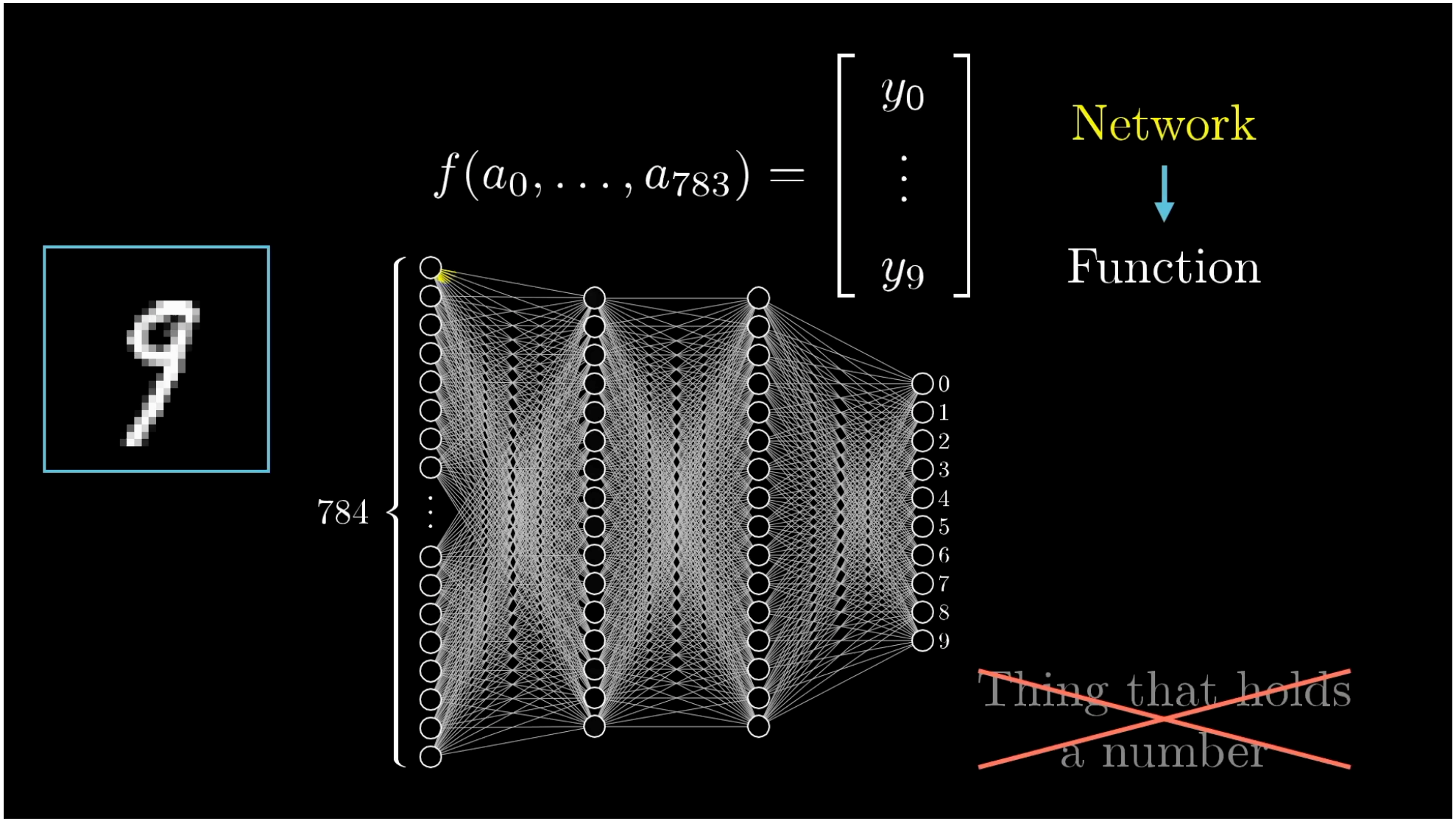
A Multi-Level Perceptron - Learning



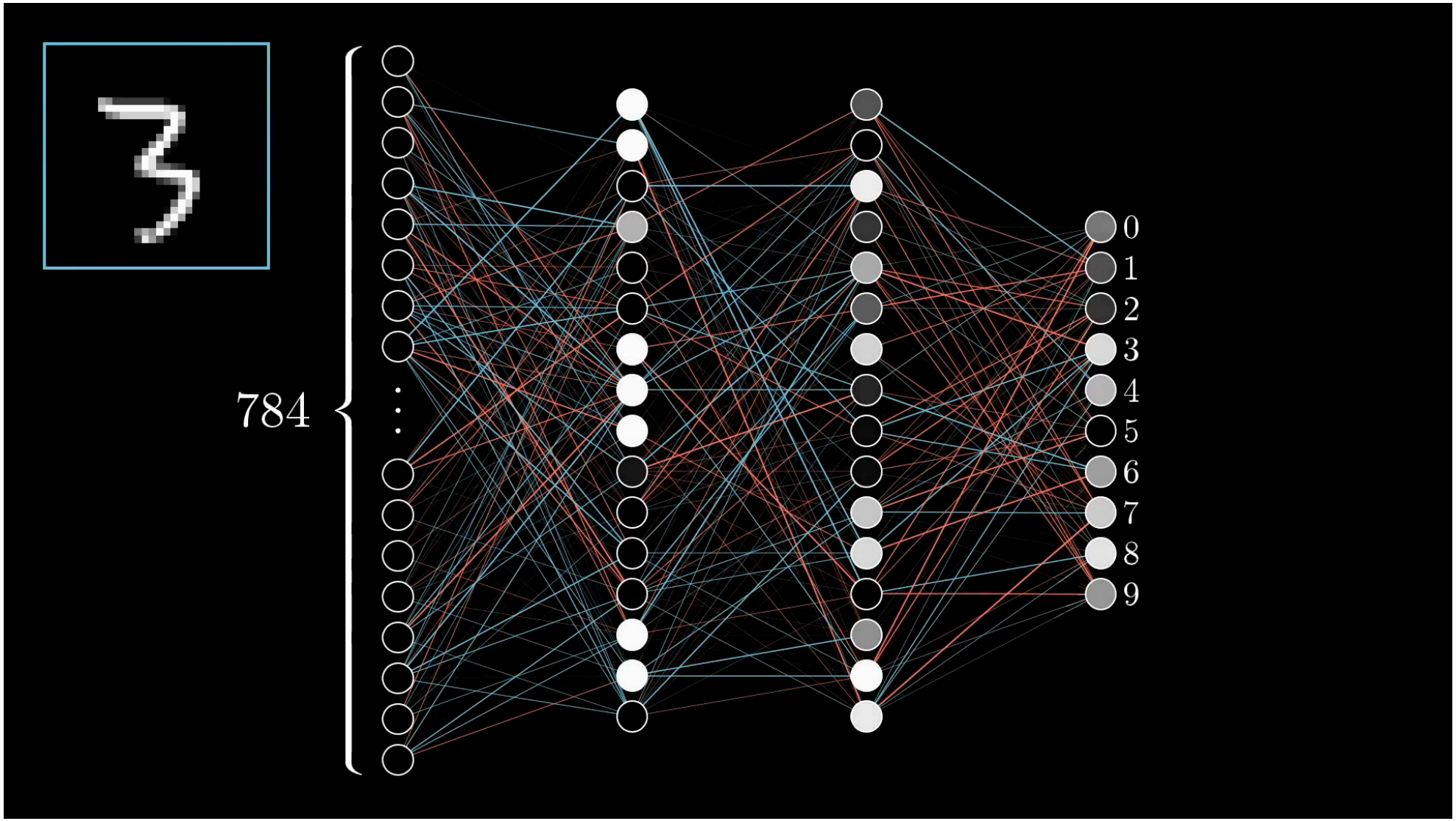
A Multi-Level Perceptron - Learning



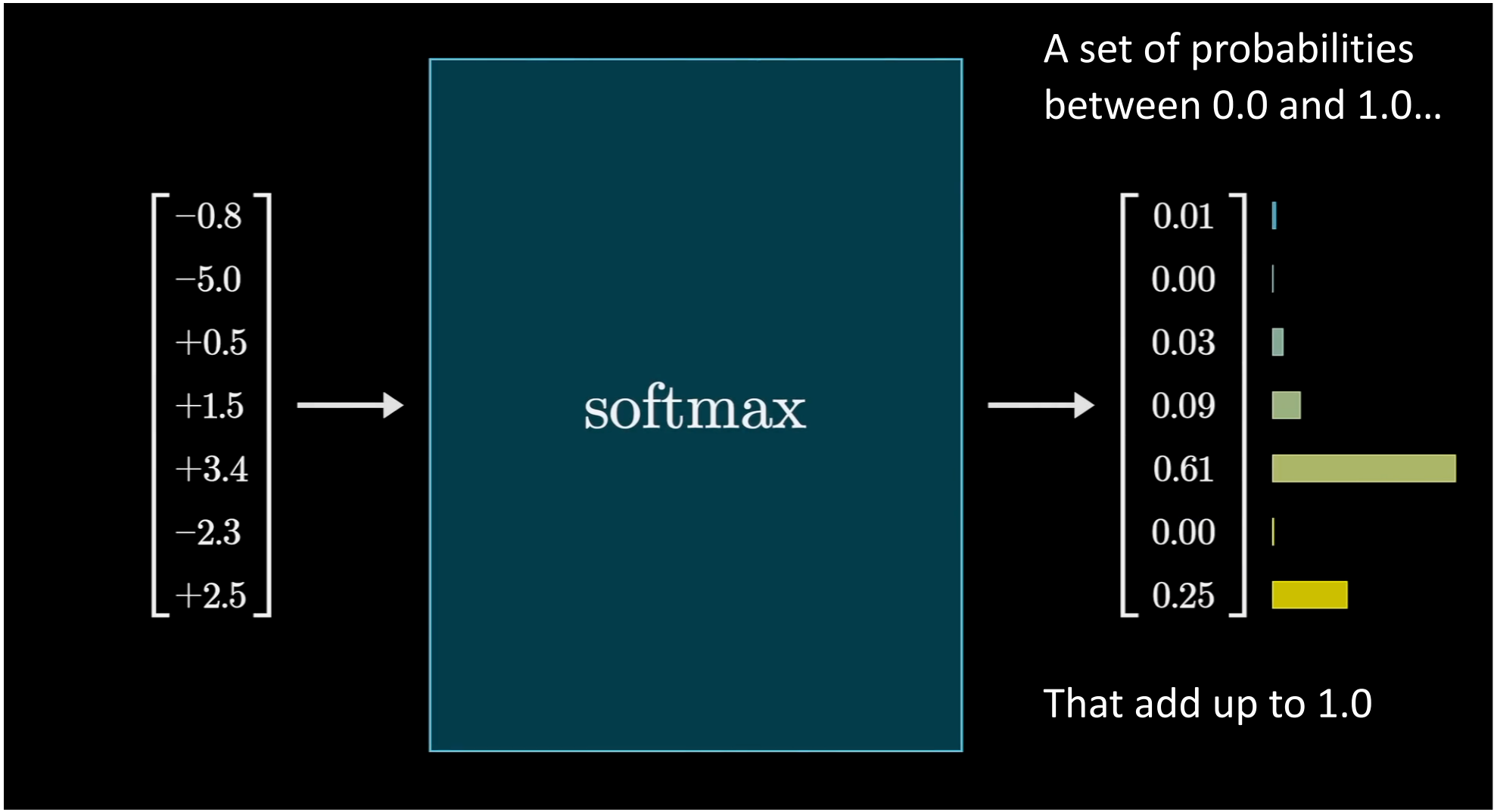
A Multi-Level Perceptron - Learning



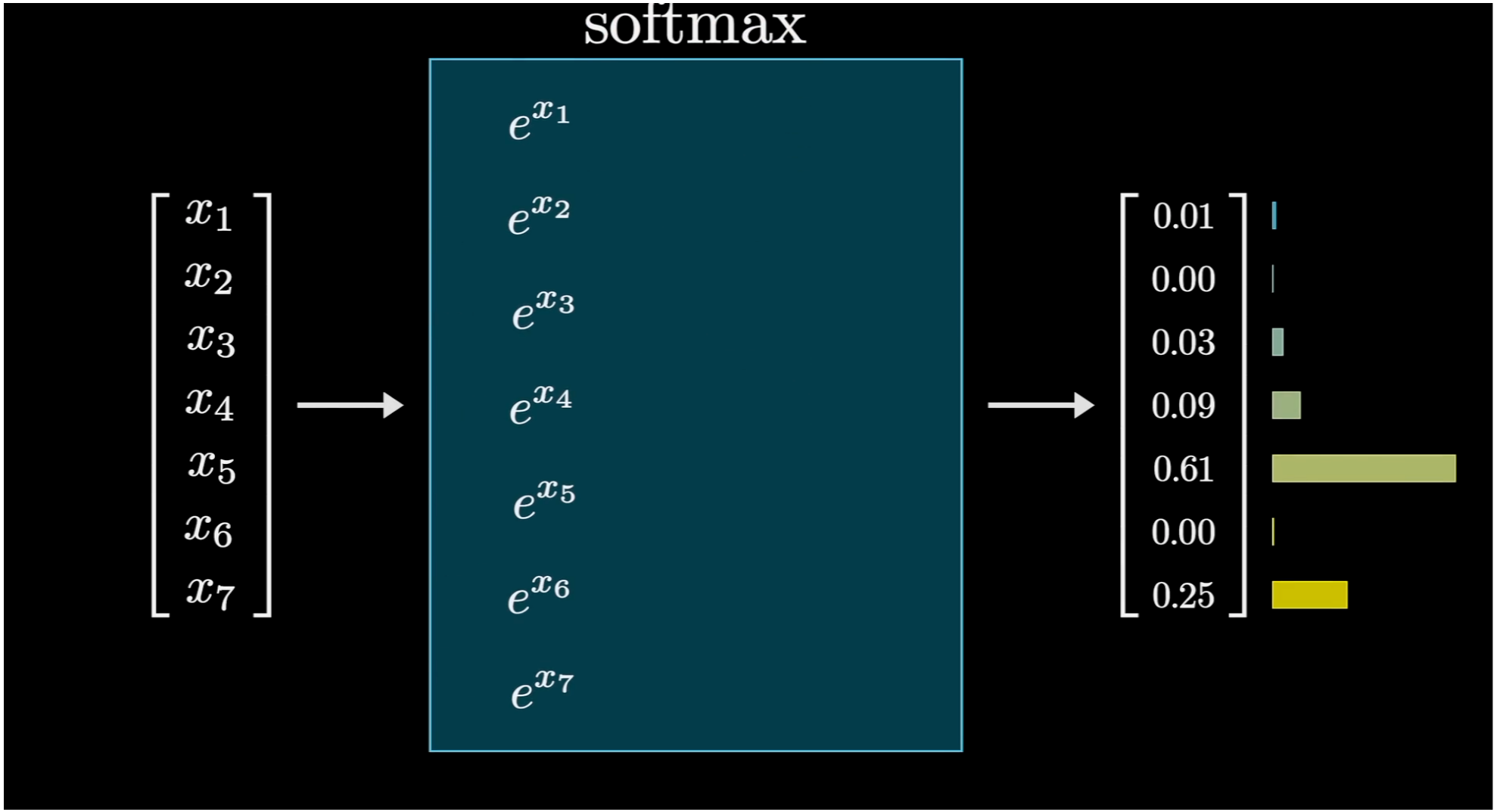
A Multi-Level Perceptron - Output



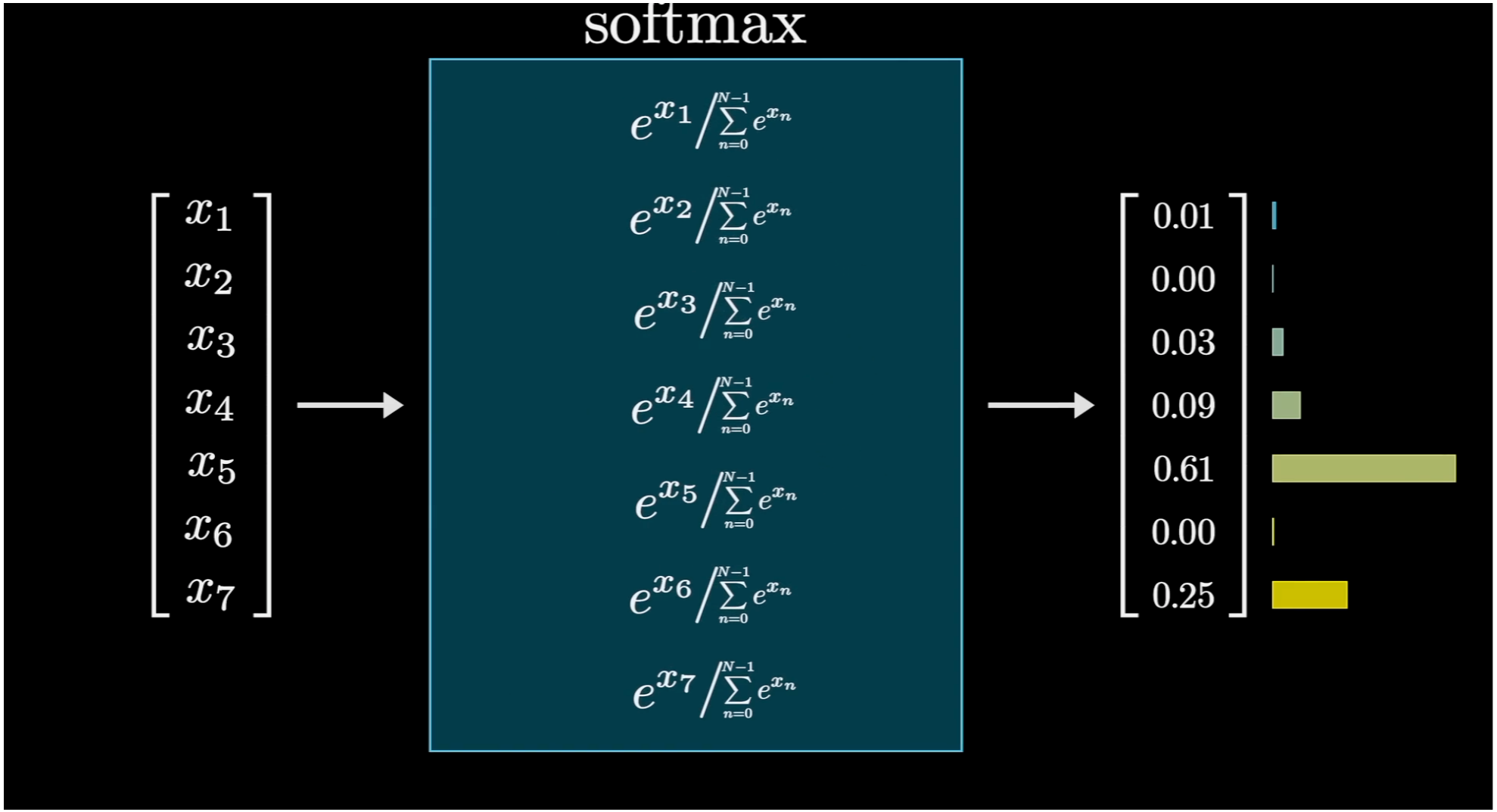
A Multi-Level Perceptron - Output



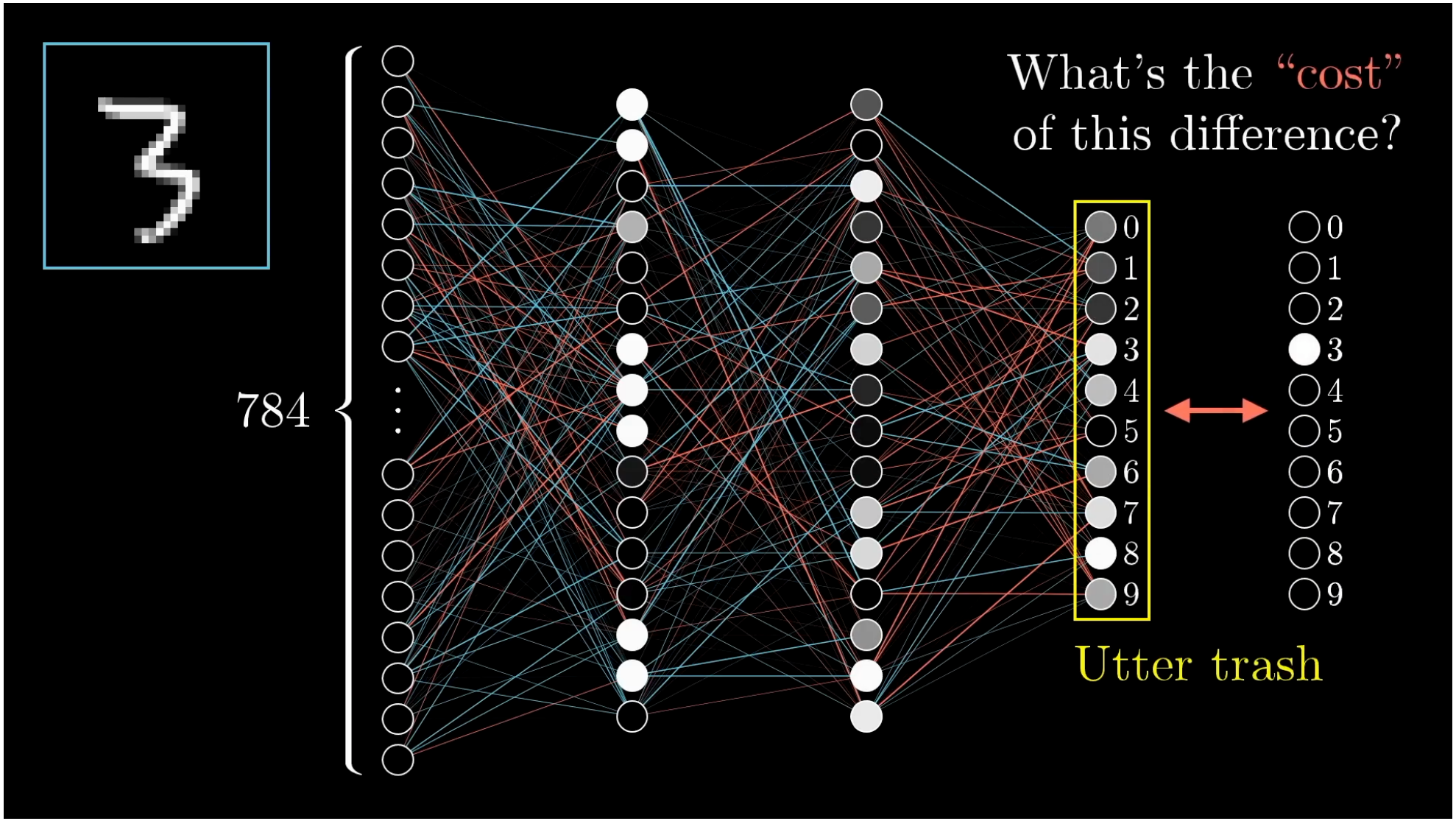
A Multi-Level Perceptron - Output



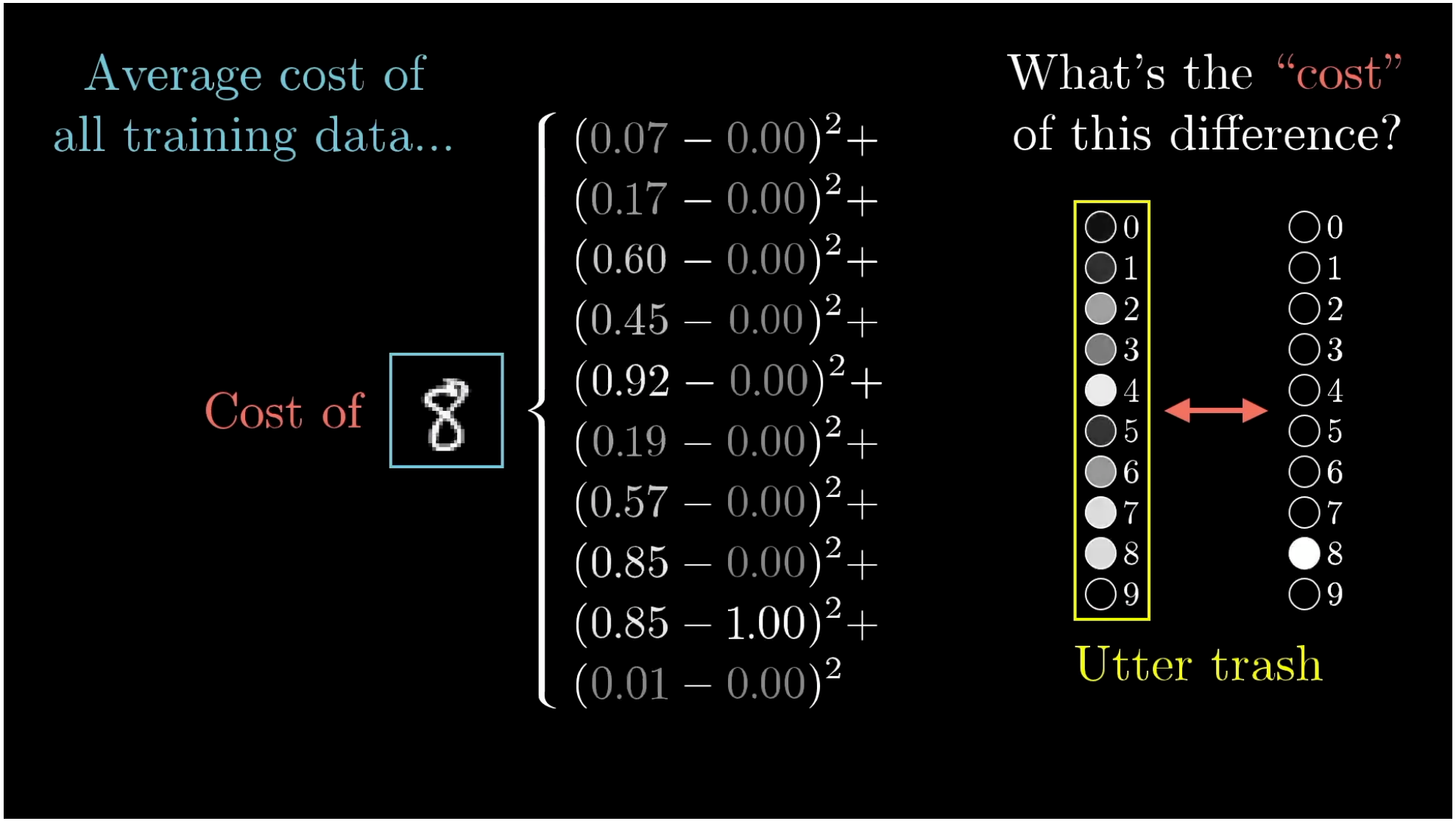
A Multi-Level Perceptron - Output



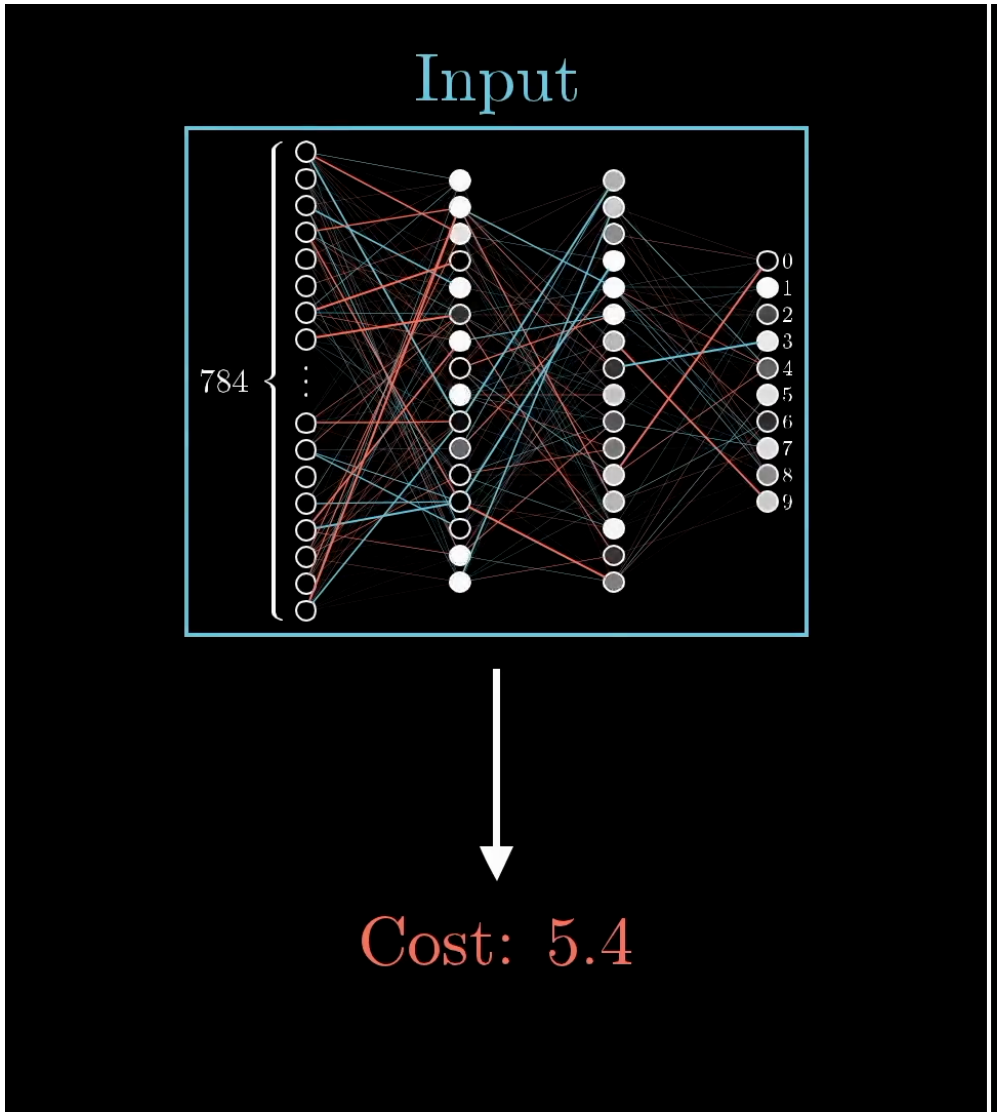
A Multi-Level Perceptron - Cost Function

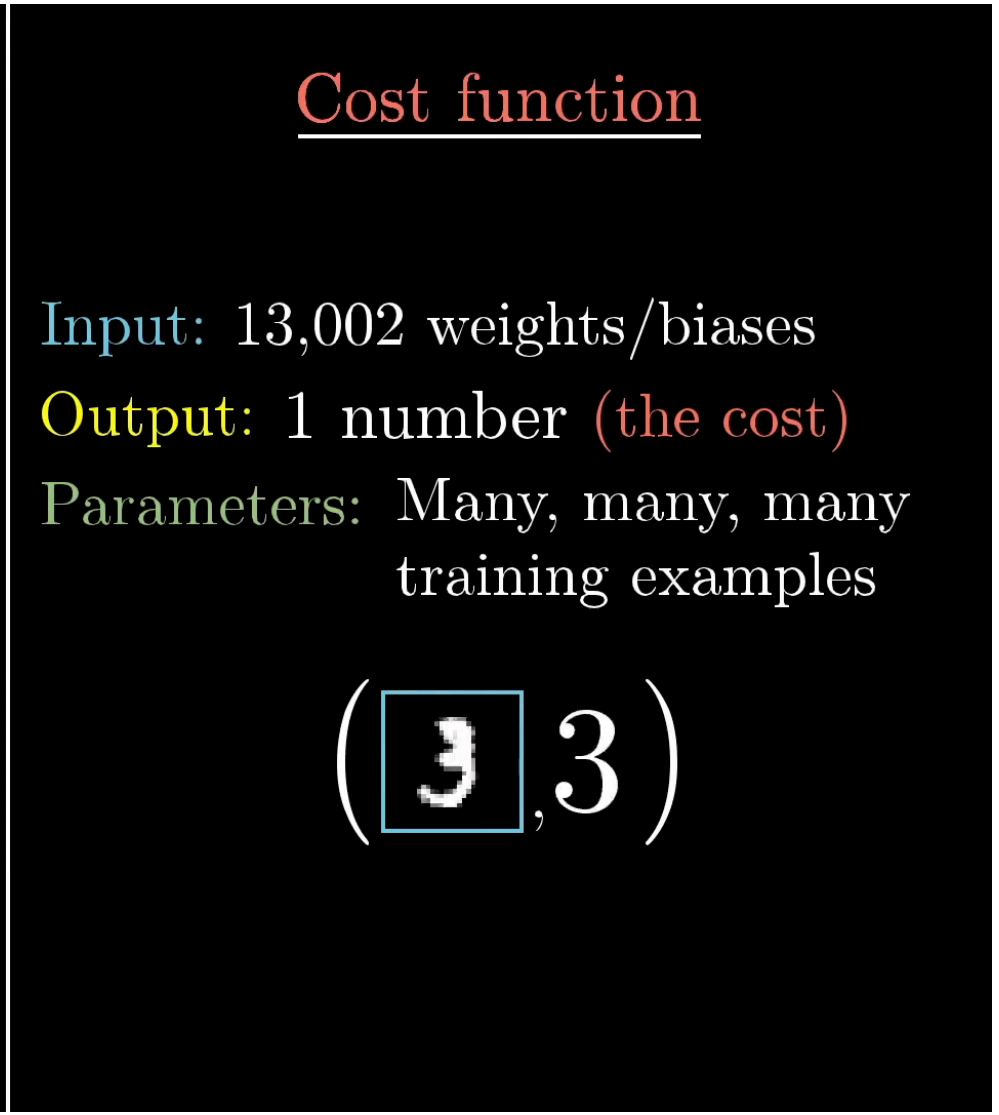


A Multi-Level Perceptron - Cost Function

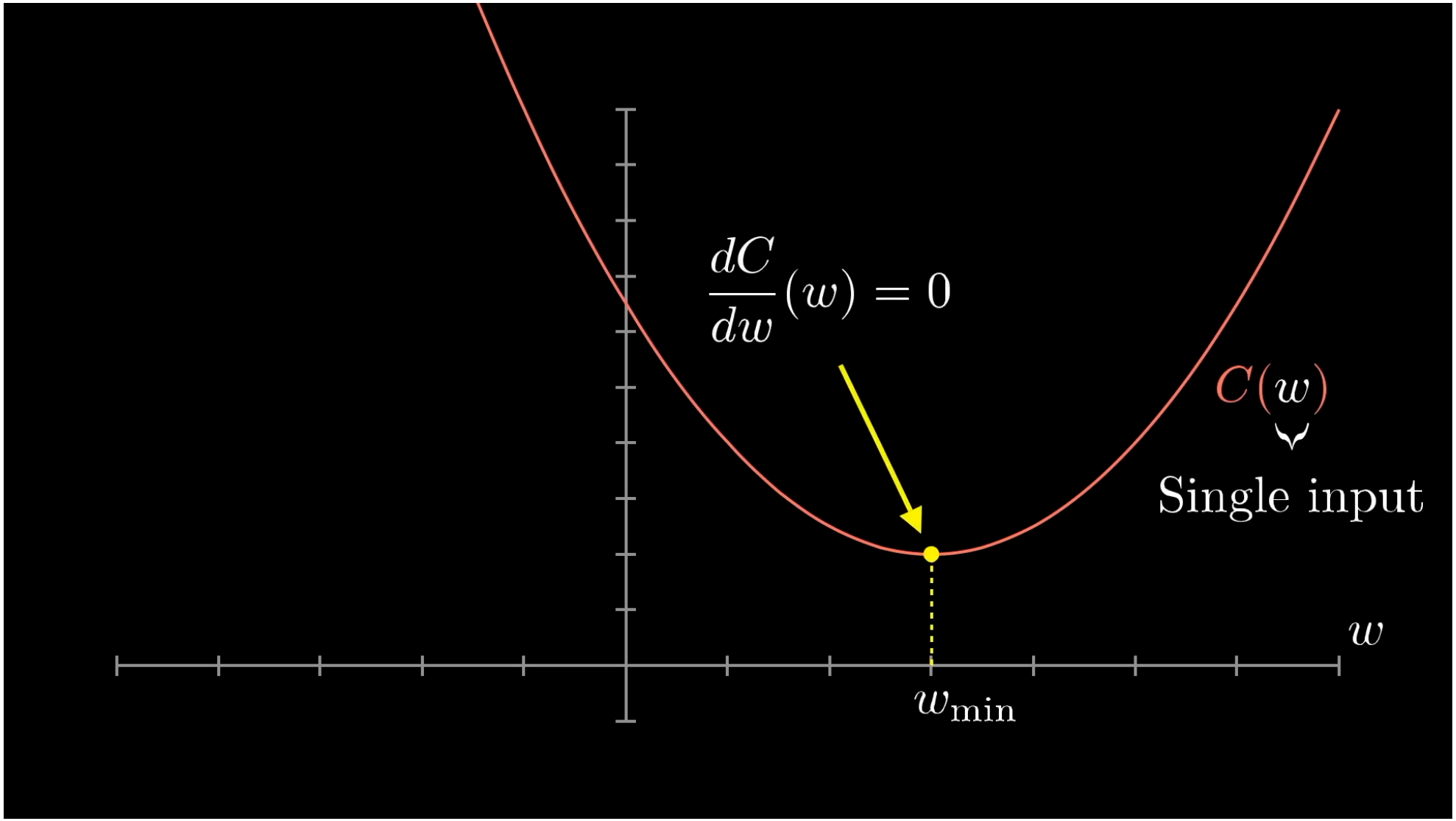


A Multi-Level Perceptron - Cost Function

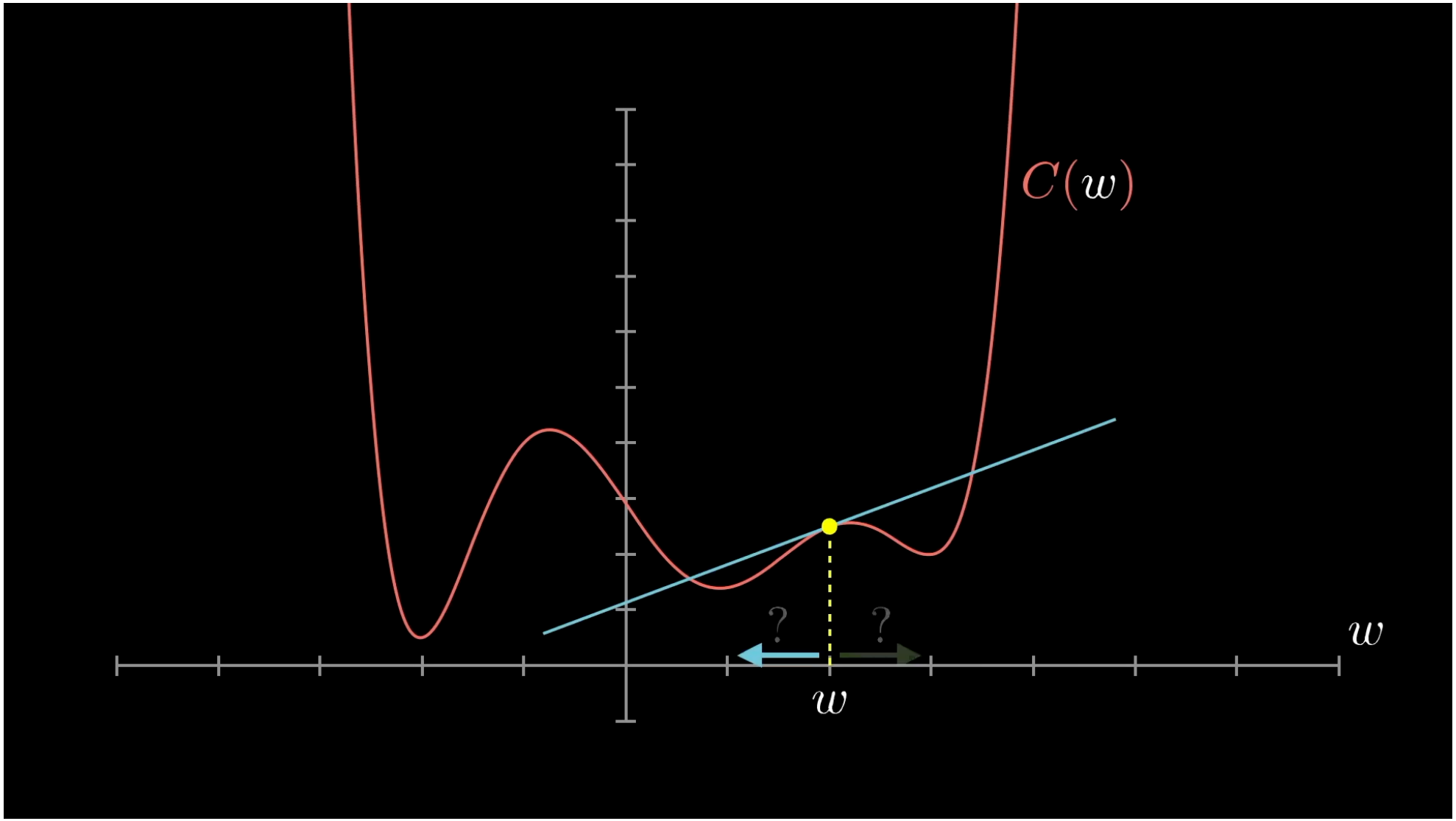




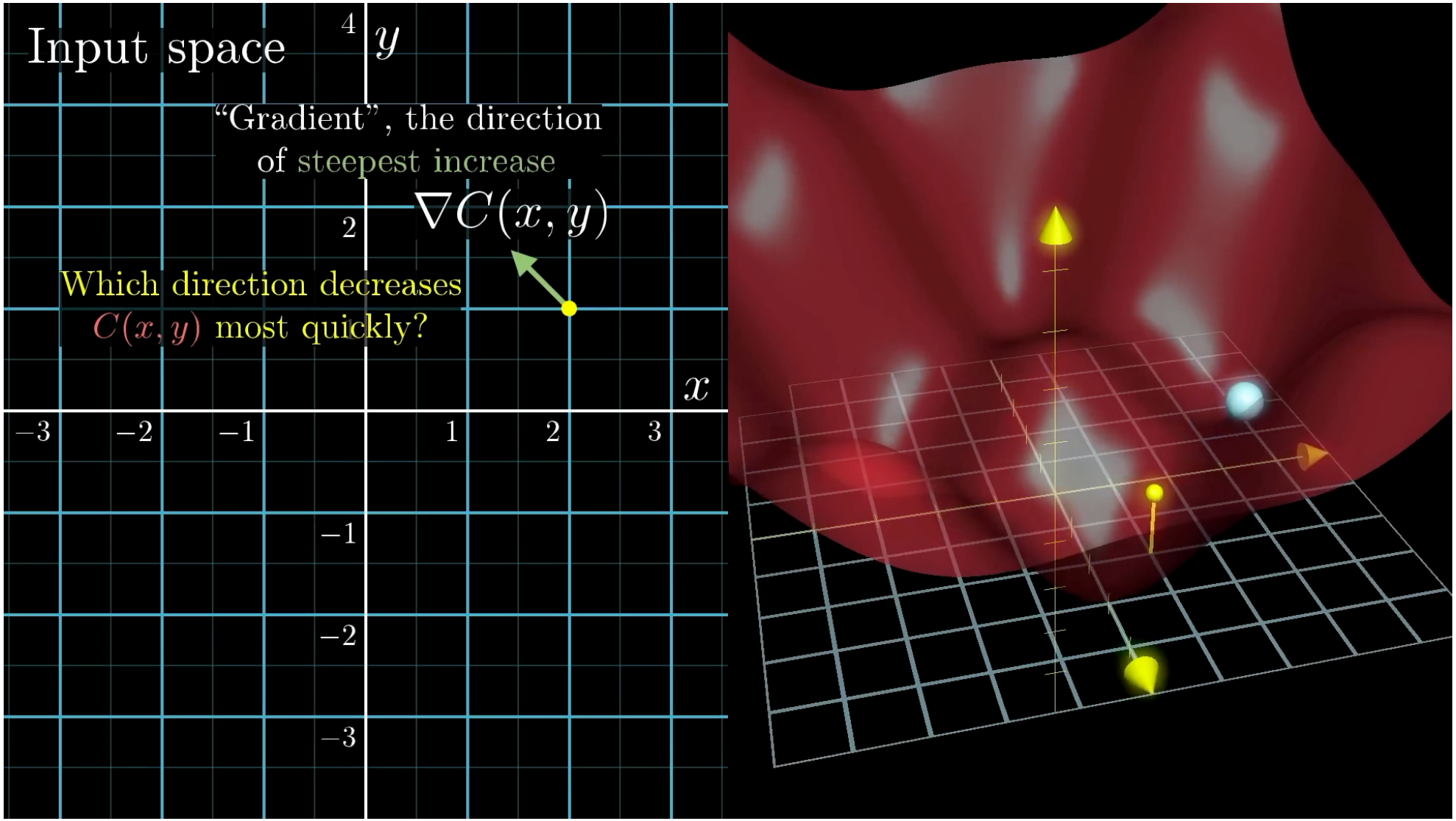
A Multi-Level Perceptron - Finding a Mimimum



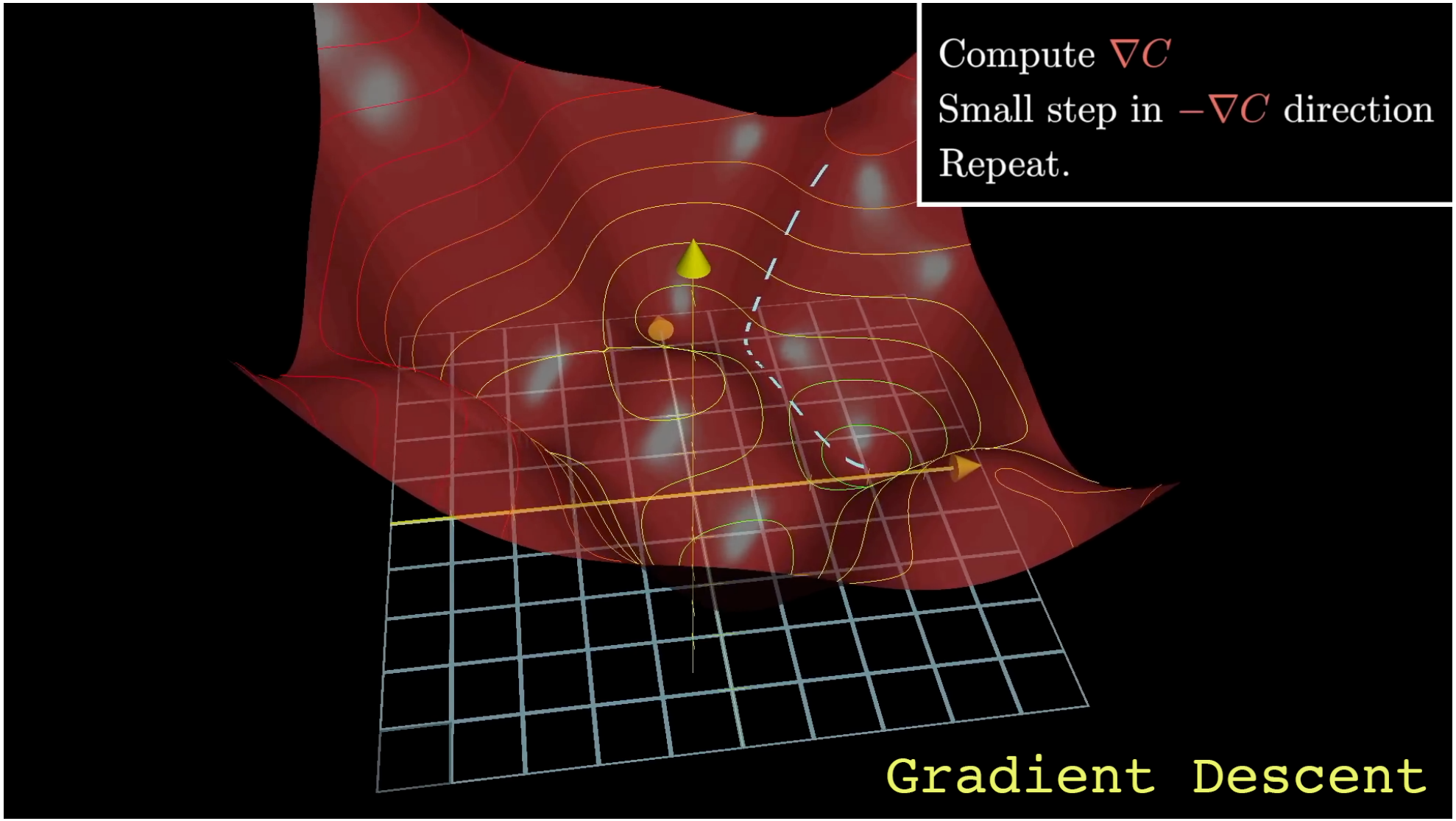
A Multi-Level Perceptron - Finding a Minimum



A Multi-Level Perceptron - Finding a Minimum



A Multi-Level Perceptron - Finding a Minimum



Backpropagation

- An algorithm for supervised learning of artificial neural networks using **gradient descent**.
- Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural network's weights and biases.
- The "backwards" part of the name stems from the fact that calculation of the gradient proceeds backwards through the network, with the gradient of the final layer of weights being calculated first and the gradient of the first layer of weights being calculated last.

Backpropagation

- **Finding w_i and b_i is very simple.**
- All we need to know is, if we tweak one variable, say w_1, would L become bigger or smaller and by how much. Then we know how to update w_i to make L smaller.

In math, it is the partial derivative $\frac{\partial L}{\partial w_i}$. Based on the chain rule, rewrite as:

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y_{pred}} * \frac{\partial y_{pred}}{\partial n_i} * \frac{\partial n_i}{\partial w_i}$$

Where each part of the above components can be computed.

Train a neural network

Step 3: Update w_1 by Stochastic Gradient Descent

Now we know if $w1$ increases by one unit, L will increase by 0.0214 unit. Or if $w1$ decreases by one unit, L will decrease by 0.0214 unit.

It's basically just this update equation: $w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1}$

Where

- η is a constant called the learning rate that controls how fast we train.
- If $\frac{\partial L}{\partial w_1}$ is positive, w_1 will decrease, which make **L** decrease.
- If $\frac{\partial L}{\partial w_1}$ is negative, w_1 will increase, which make **L** increase.
- If we do this for every weight and bias in the network, the loss will slowly decrease and our network will improve.

Train a neural network

Step 4: repeat step 2,3 to update all *w* and *b* for this sample

Step 5: repeat step 1,2,3,4 for all training samples

- Does the backpropagation algorithm always find the optimal w and b to have the global minimum of L?
  - The main challenge is to iteratively update our initial guess so that we find the right set of parameters that converge to the minimum of the loss landscape.
  - One best practice to tuning the learning rate is to initially set it to a relatively high value like 0.1 to stall the network resulting in large loss, and then successively halve the learning rate until the network begins to show signs of convergence.