From a Single Neuron to a Full Neural Network

Recap: The Neuron

A single neuron takes:

$$y = f(w \cdot x + b)$$

Where:

- x = input vector
- w = weights
- b = bias
- f = activation function

What's a layer?

A layer is just a group of neurons that process the same input in parallel

If we have:

- n_{inputs} = # Features in x
- $n_{neurons}$ = # neurons in layer

Then:

- Weights: A matrix W of shape $(n_{inputs}, n_{neurons})$
- Biases: A vector of length $n_{neurons}$

Forward pass for a layer:

$$z = xW + b$$

$$y = f(z)$$

Where:

- x is a row vector for one sample (or a matrix for batches)
- Multiplication is matrix multiplication
- The activation f is applied elementwise

Example:

If:

- $x = [x1, x2, x3]$
- 2 neurons in layer → W is 3×2 matrix, b is length 2 vector
- Output: $[y1, y2]$

From Layer to Network

A network is just stacked layers:

Input → Layer 1 → Layer 2 → ... → Output Layer

For 3 layers:

$$a^{(1)} = f_1(xW^{(1)} + b^{(1)})$$
$$a^{(2)} = f_1(a^{(1)}W^{(2)} + b^{(2)})$$
$$a^{(3)} = f_3(a^{(2)}W^{(3)} + b^{(3)})$$

Python Implementation in corresponding notebook

How LLMs are Built From This

Every modern neural network - CNN, RNN, Transformer - is just a specialized way of connecting layers of neurons

The forward pass math is always:

$$\text{output} = \text{activation}(\text{input} \times W + b)$$

...repeated many times w/ diff architectural tweaks