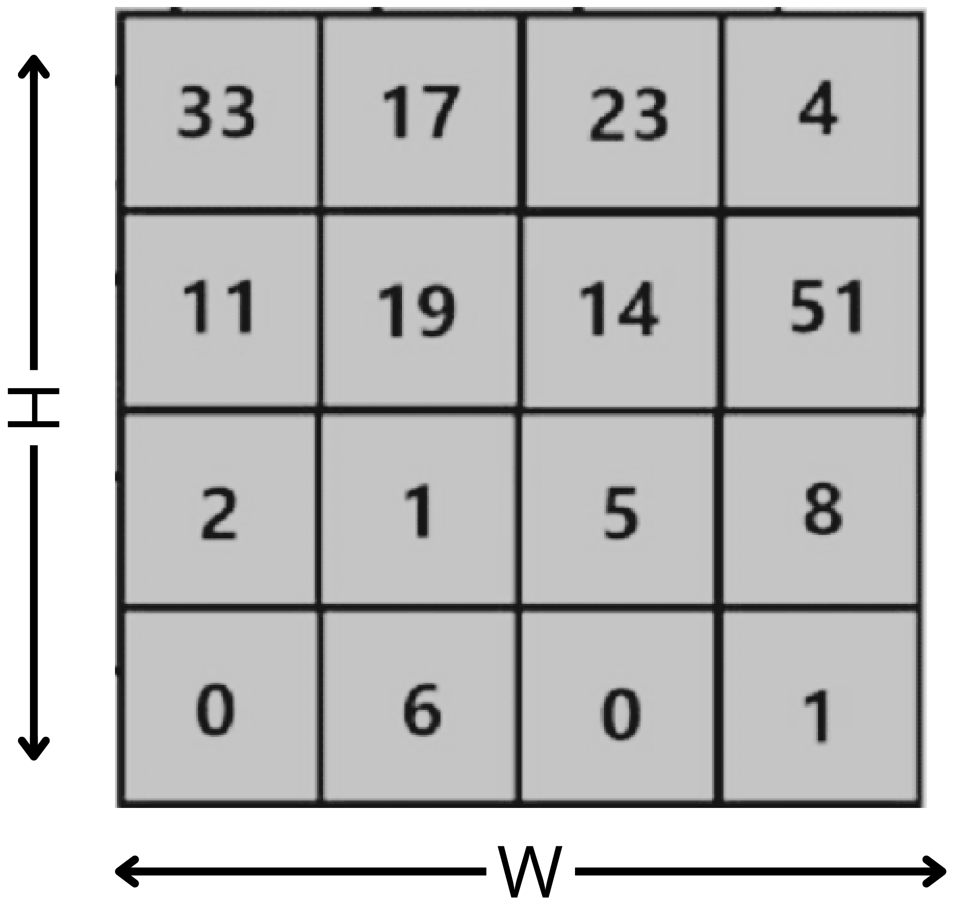
Applied mathematics in Deep Learning

Picture to numbers

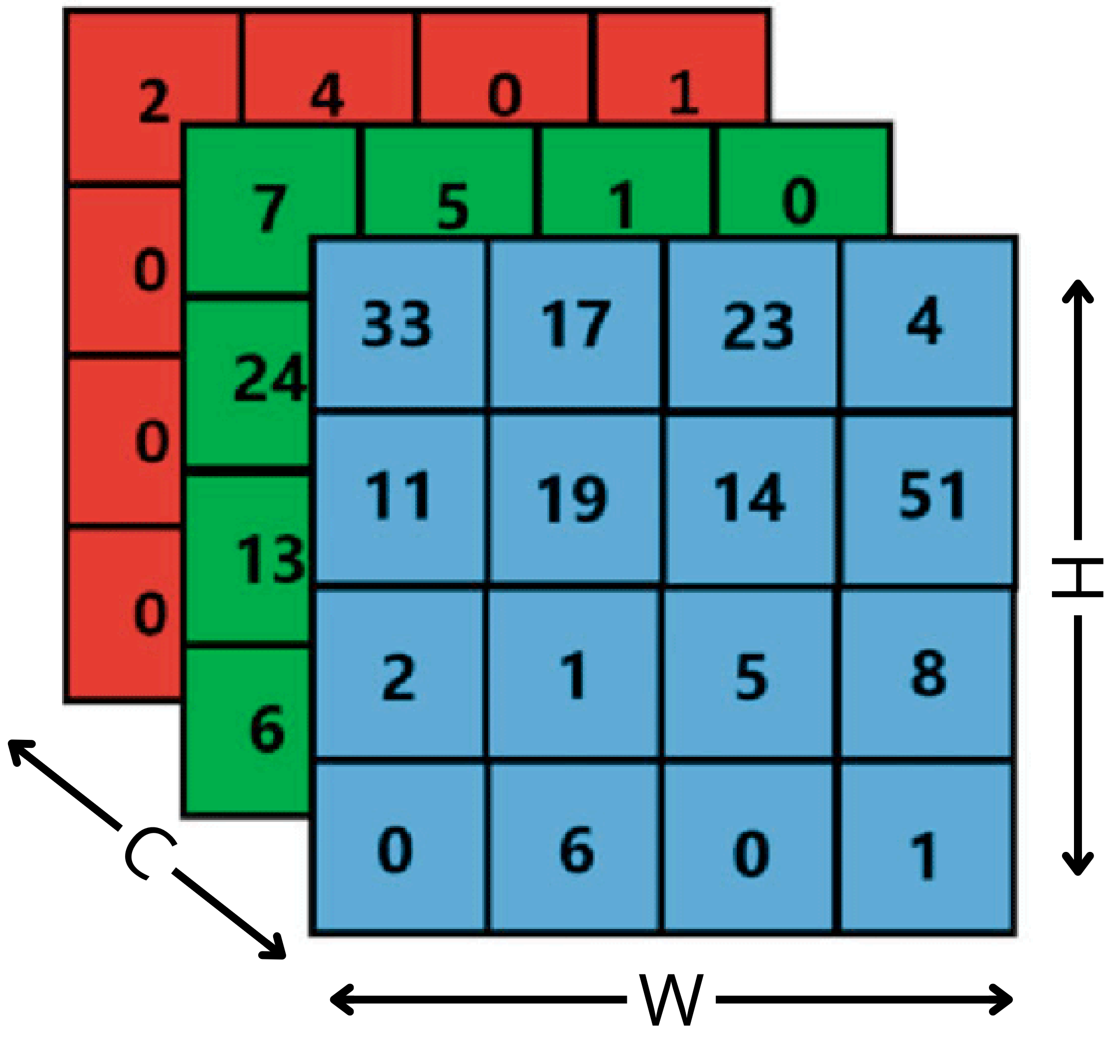
What is the input format in a deep learning network?

Grayscale image = vector = H X W



33	17	23	4
11	19	14	51
2	1	5	8
0	6	0	1

RGB image = vector = C x H x W



33	17	23	4
11	19	14	51
2	1	5	8
0	6	0	1

Picture to numbers

What is the input format in a deep learning network?

$$X \in \mathbb{R}^n$$

Where

- **X** contains the pixel values of an image
- $\in$ means it’s an “Element of”
- $\mathbb{R}^n$ means "a vector with n real numbers."

So we say: “**X** is a vector with **n** numbers representing pixel values.”

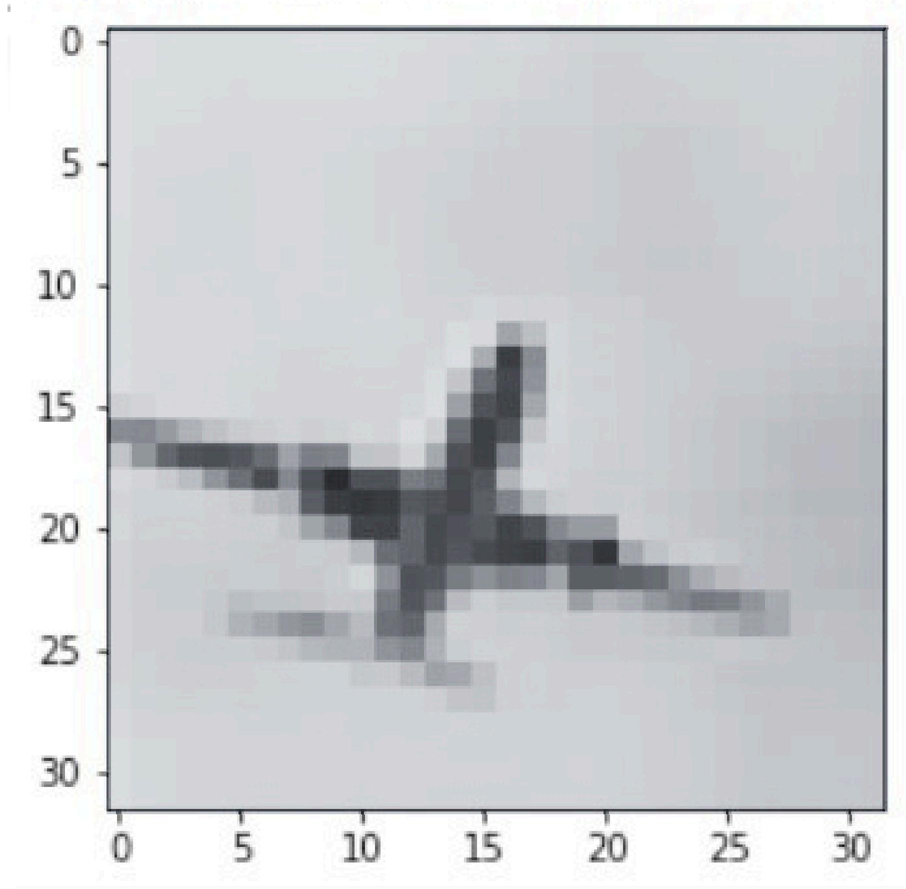
Picture to numbers

What is the input format in a deep learning network?

Example 32x32 image

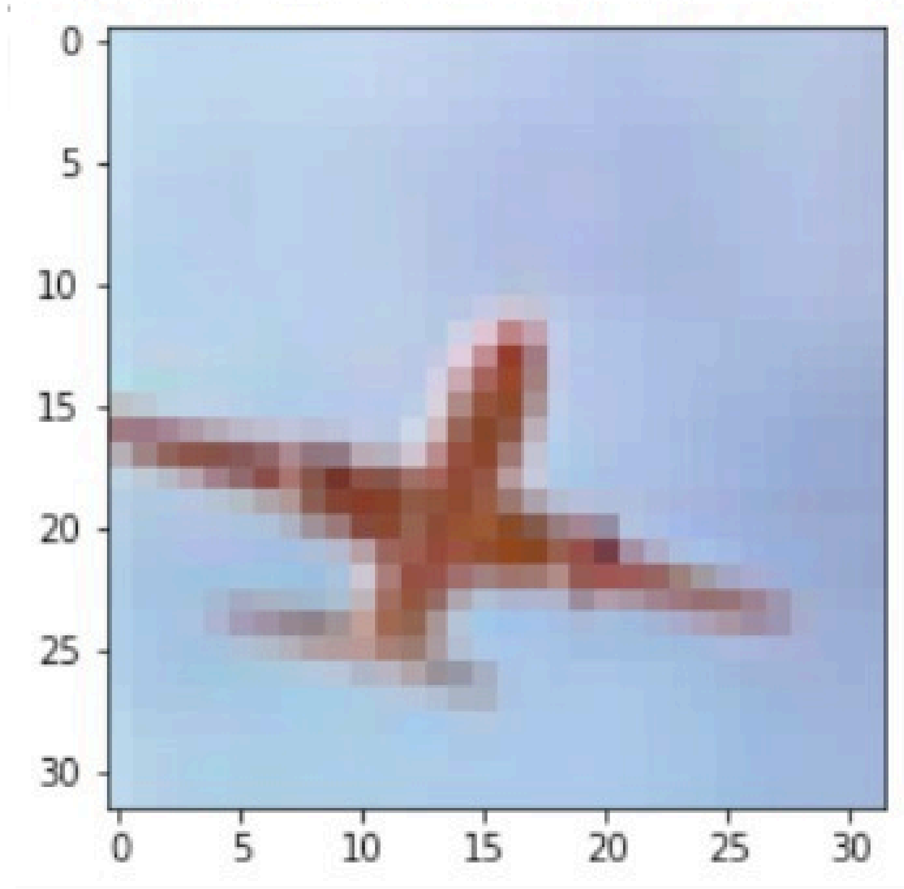
For **grayscale**

32 x 32 = 1024 pixels = $x \in \mathbb{R}^{1024}$



For **RGB**

32 x 32 x 3 = 3072 pixels = $x \in \mathbb{R}^{3072}$



Weights & Biases

How the network learns

Weights (W) - How important is each input?

Biases (B) - How important is it to respond to an input

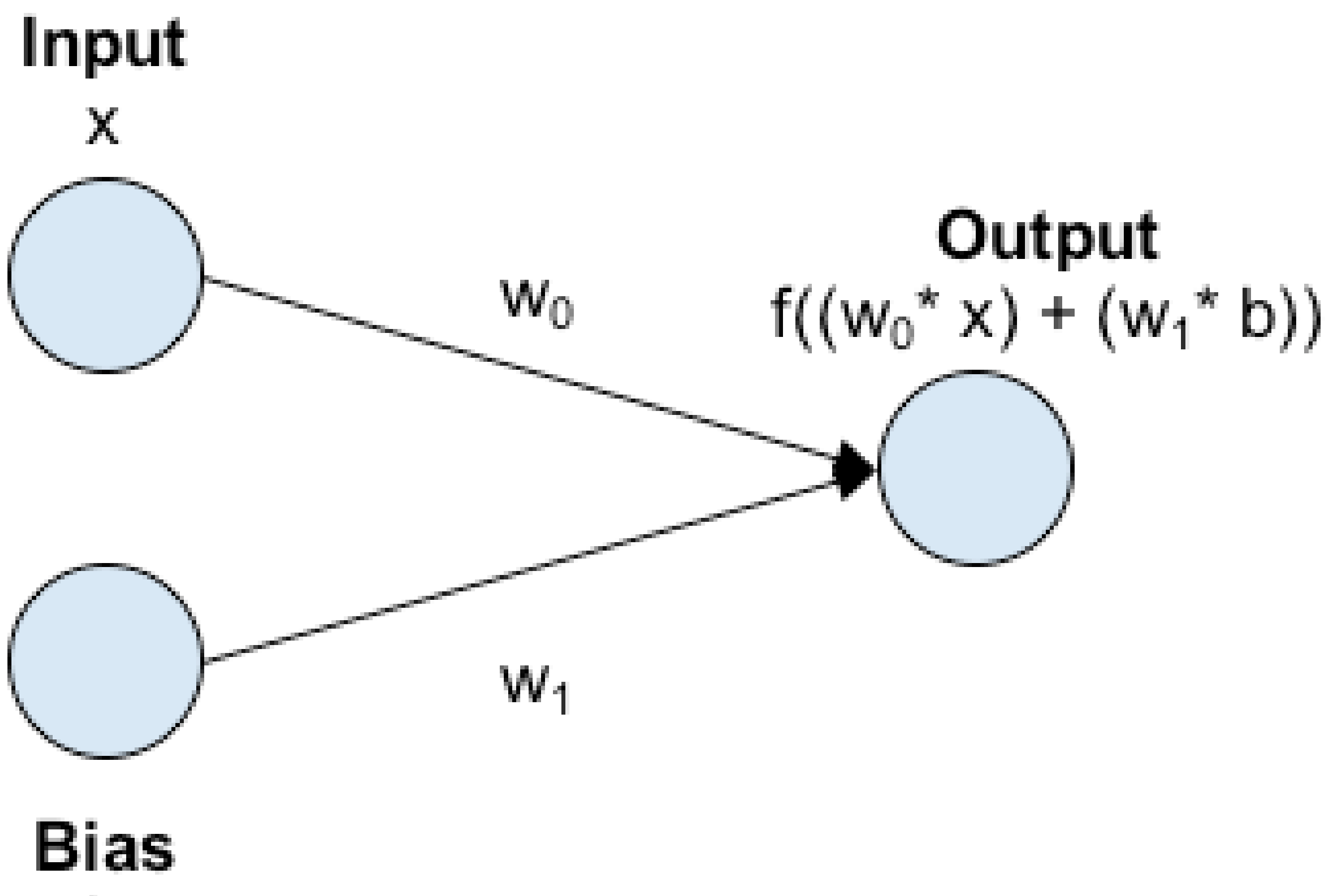
For a 'forward pass' in a network for a single input:

Output = f ((w_0 * x) + (w_1 * b))

Where

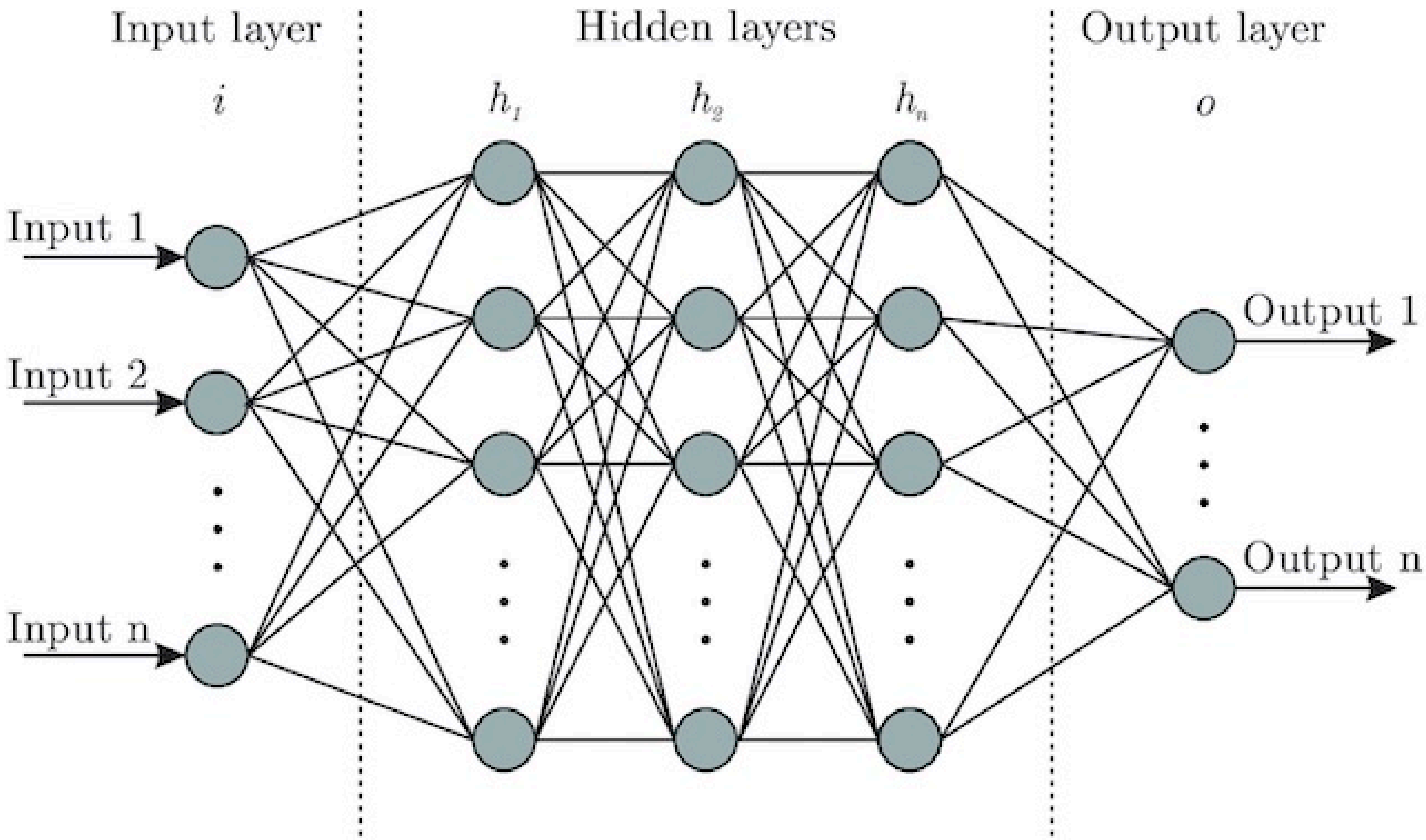
- $\mathbf{w_0}$ controls how important the input is
- $\mathbf{w_1}$ shifts the output results
- **f** is the activation function

Bias



Layered architecture

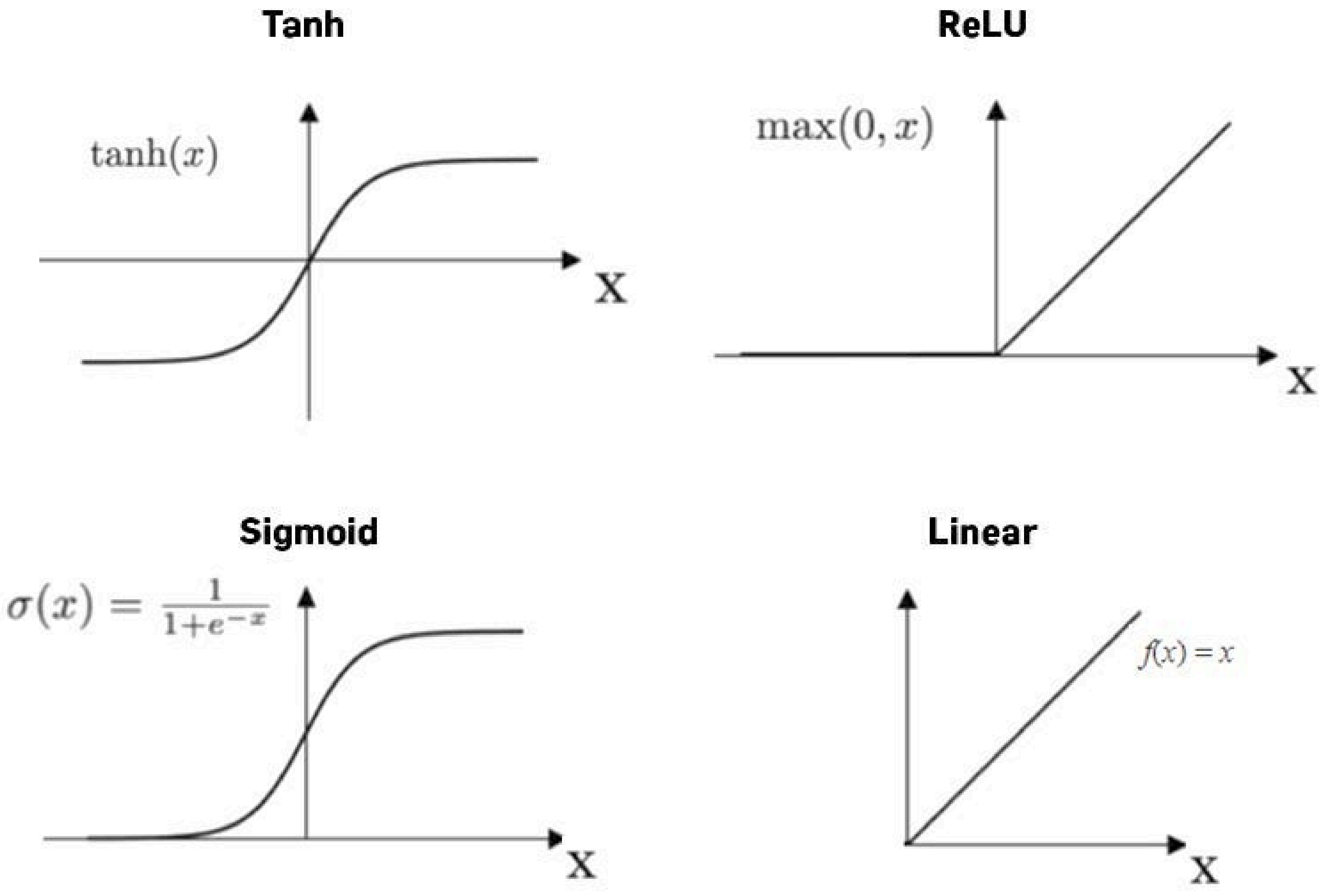
Input, output and hidden layers



Activation functions

Add non linearity

Activation function: decides the output of the neuron

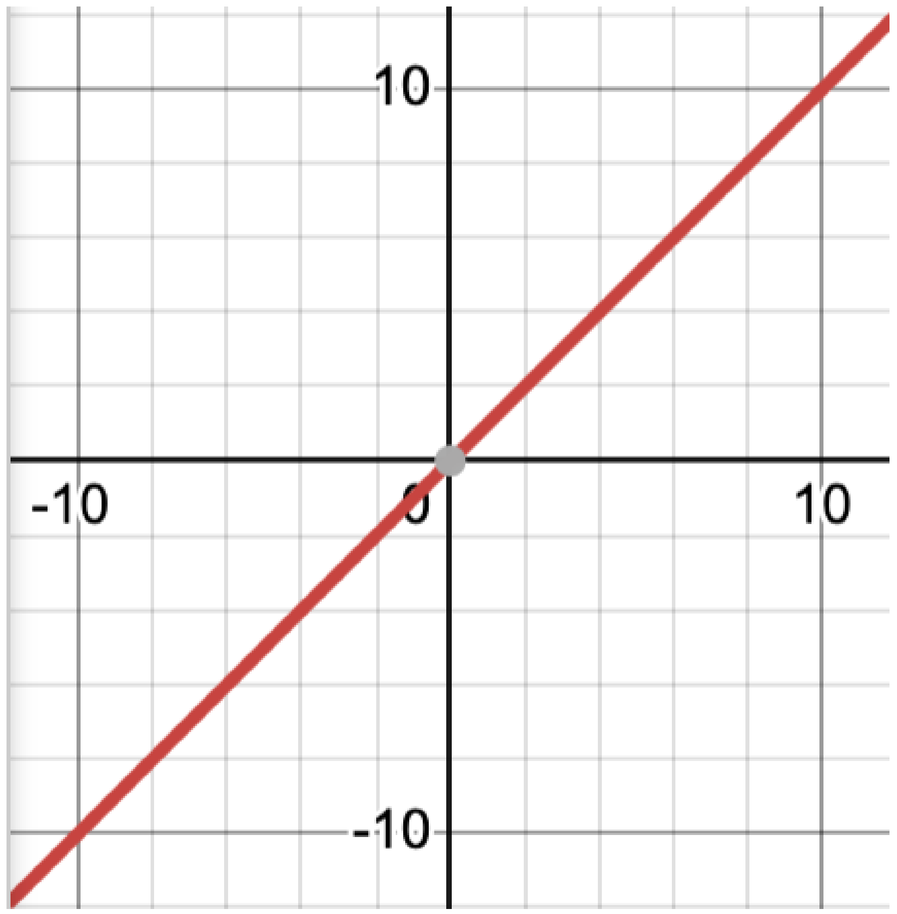


Activation functions: Linear functions

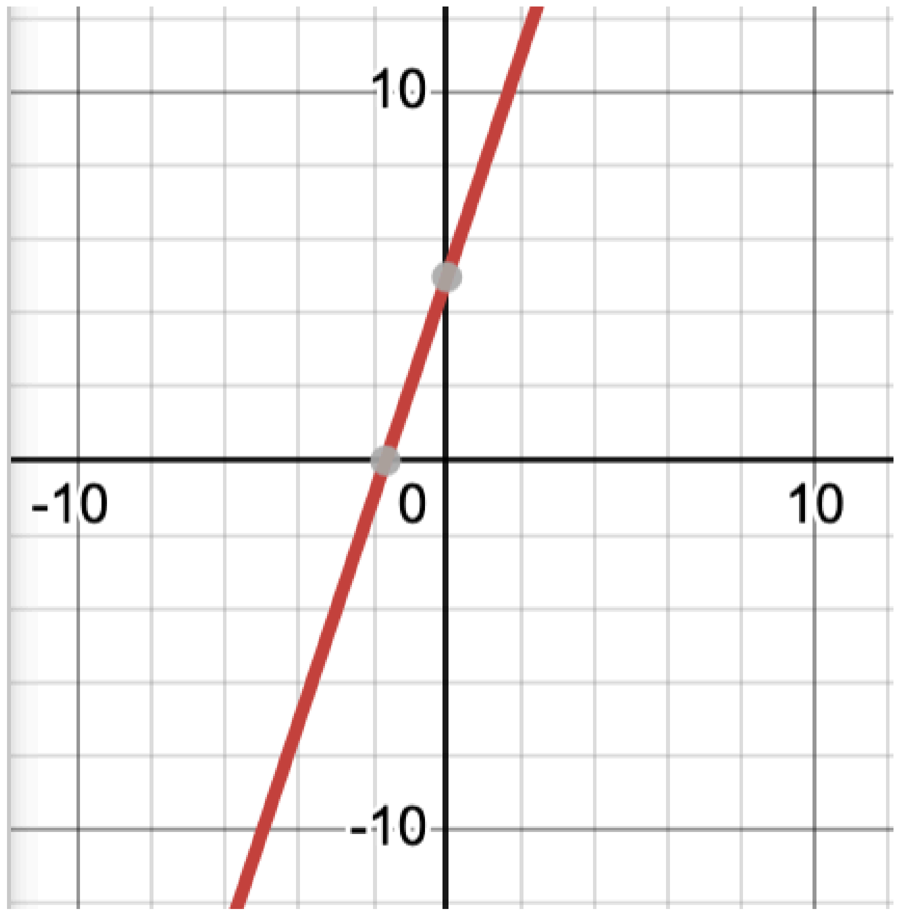
A linear function means:

- If you double the input, the ouput doubles too
- If you add two inputs, they are added too the output too

f(x) = x + 0



f(x) = 3x + 5



Activation functions: ReLU

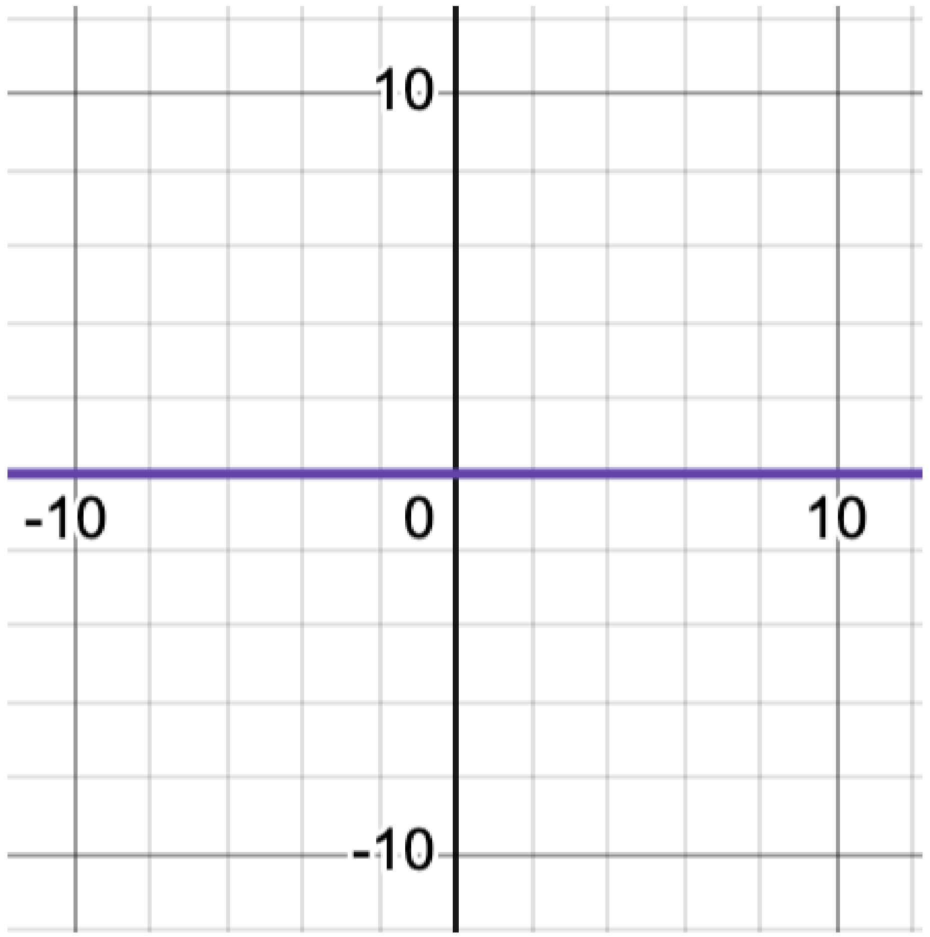
ReLU (Rectified Linear Unit): helps negative neurons ‘turn off’

$f(x) = \max(0, x)$

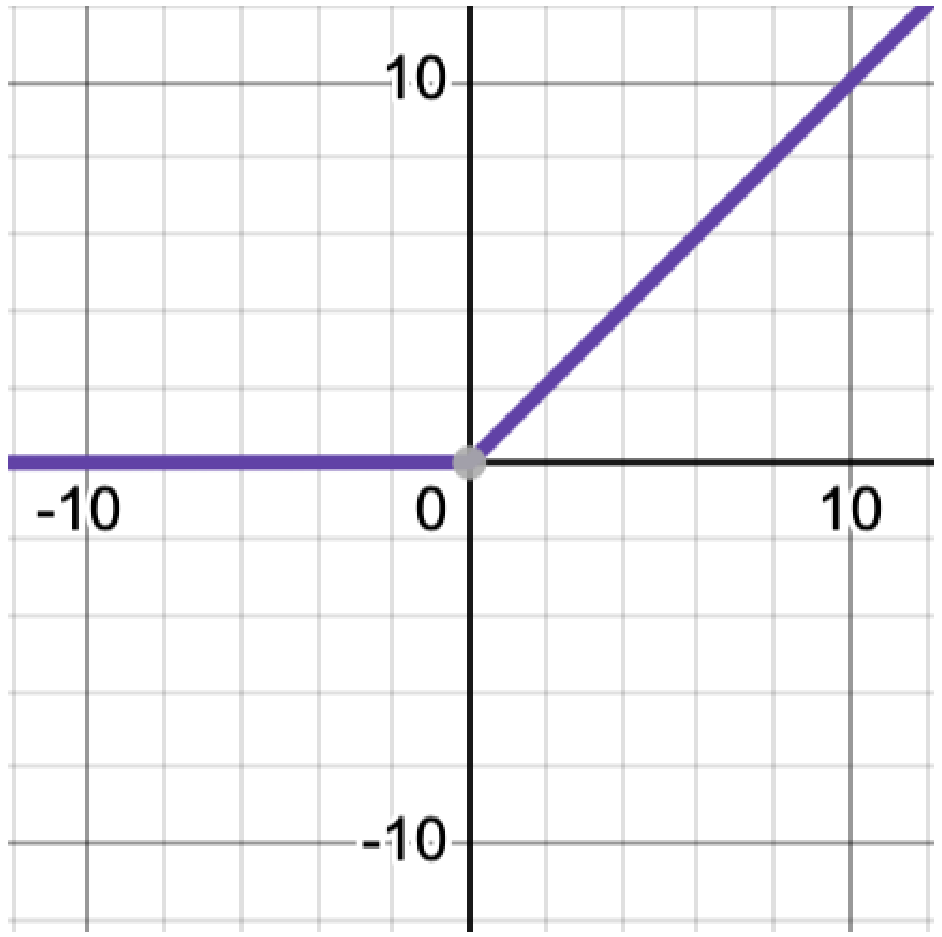
Where:

- For $x < 0$: the output is always 0 → flat line
- For $x \geq 0$: the output equals x → diagonal line going up

$x < 0$



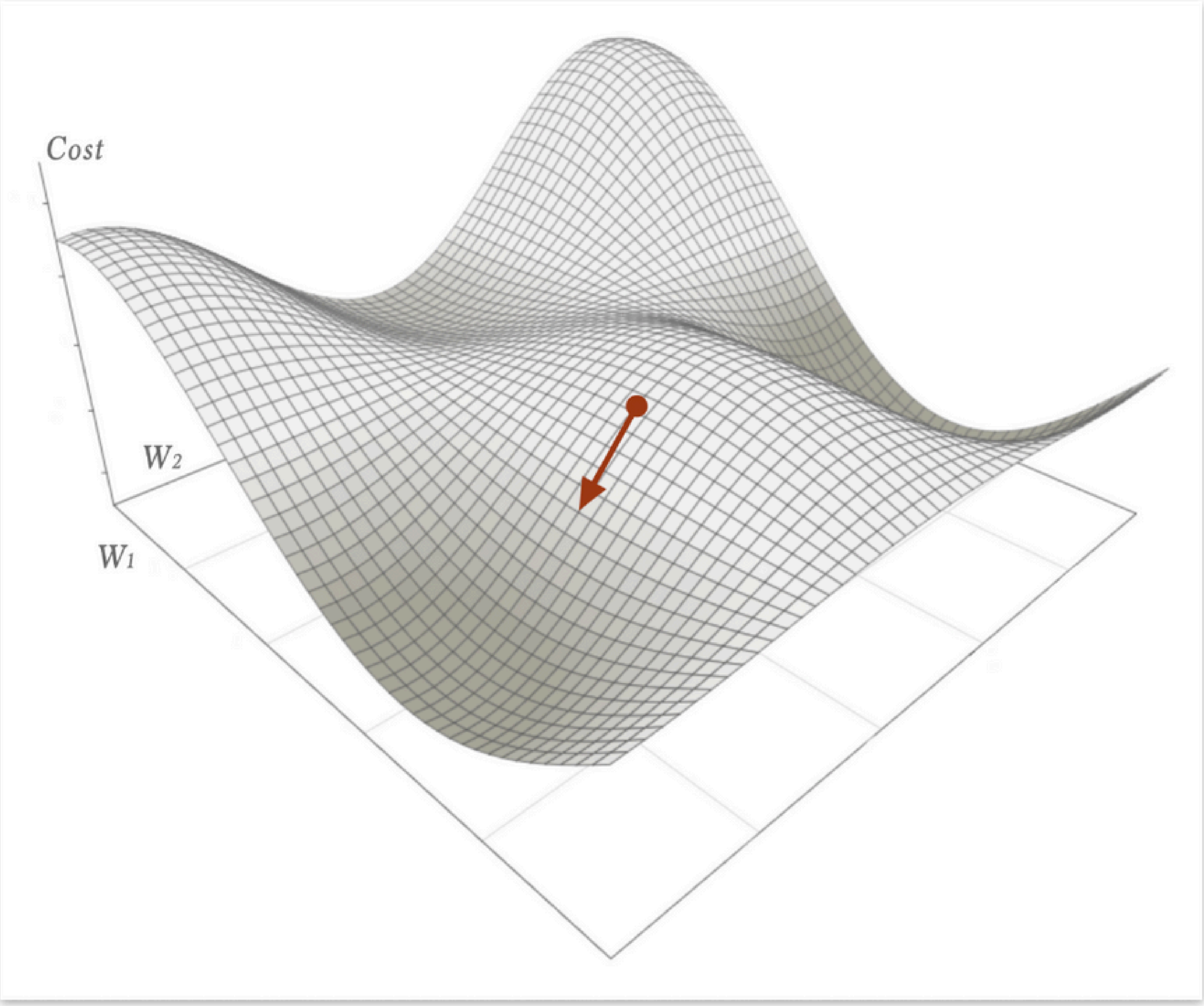
$x \geq 0$



Loss landscape

“How much is the network wrong per step?”

- If the model is confident and correct, the loss is **small**.
- If the model is wrong or unsure, the loss is **big**.



Loss landscape: Cross Entropy (CE)

Used when classes are balanced

$$CE_i = -\log(\text{softmax}(z_i)[y_i])$$

Where

- z_i: the model outputs (numbers)
- y_i: the correct class (index)
- softmax(z_i)[y_i]: the predicted probability for the correct class

Loss landscape: Focal Loss

$$FL_i = \alpha(1 - p_i)^\gamma \cdot (-\log(\mathrm{softmax}(z_i)[y_i]))$$

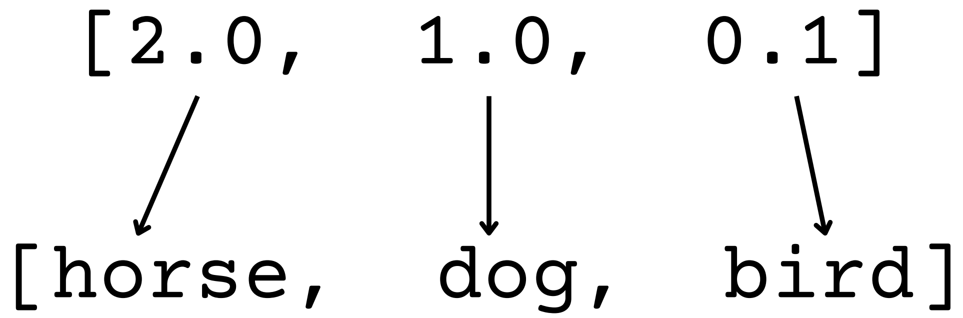
Where

- **α** balances the contribution between positive and negative classes.
- **γ** controls the value of weights for harder to classify examples
- **p** the predicted probability

Making a prediction: Logits

3 classes: horse, dog, bird.

Final layer output: [2.0, 1.0, 0.1]



[2.0, 1.0, 0.1]

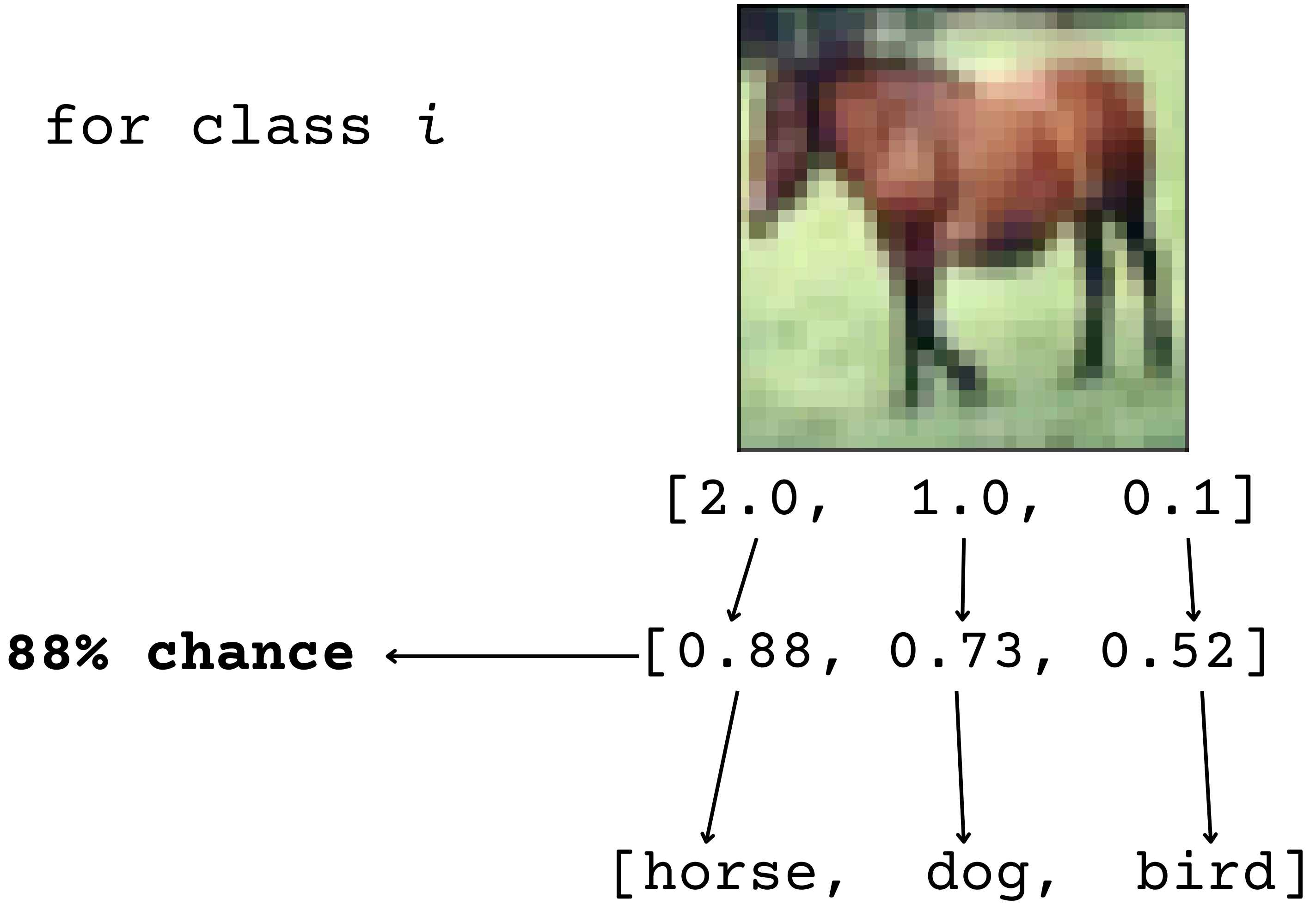
[horse, dog, bird]

Making a prediction: Softmax

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\text{sum of } e^{\text{all } x_j}}$$

Where

- x_i the score of the class i (probability)
- x_j the scores (logits) of all classes
- e^{x_i} the exponential (raising a number to a power) for class *i*



[2.0, 1.0, 0.1]

88% chance ← [0.88, 0.73, 0.52]

[horse, dog, bird]