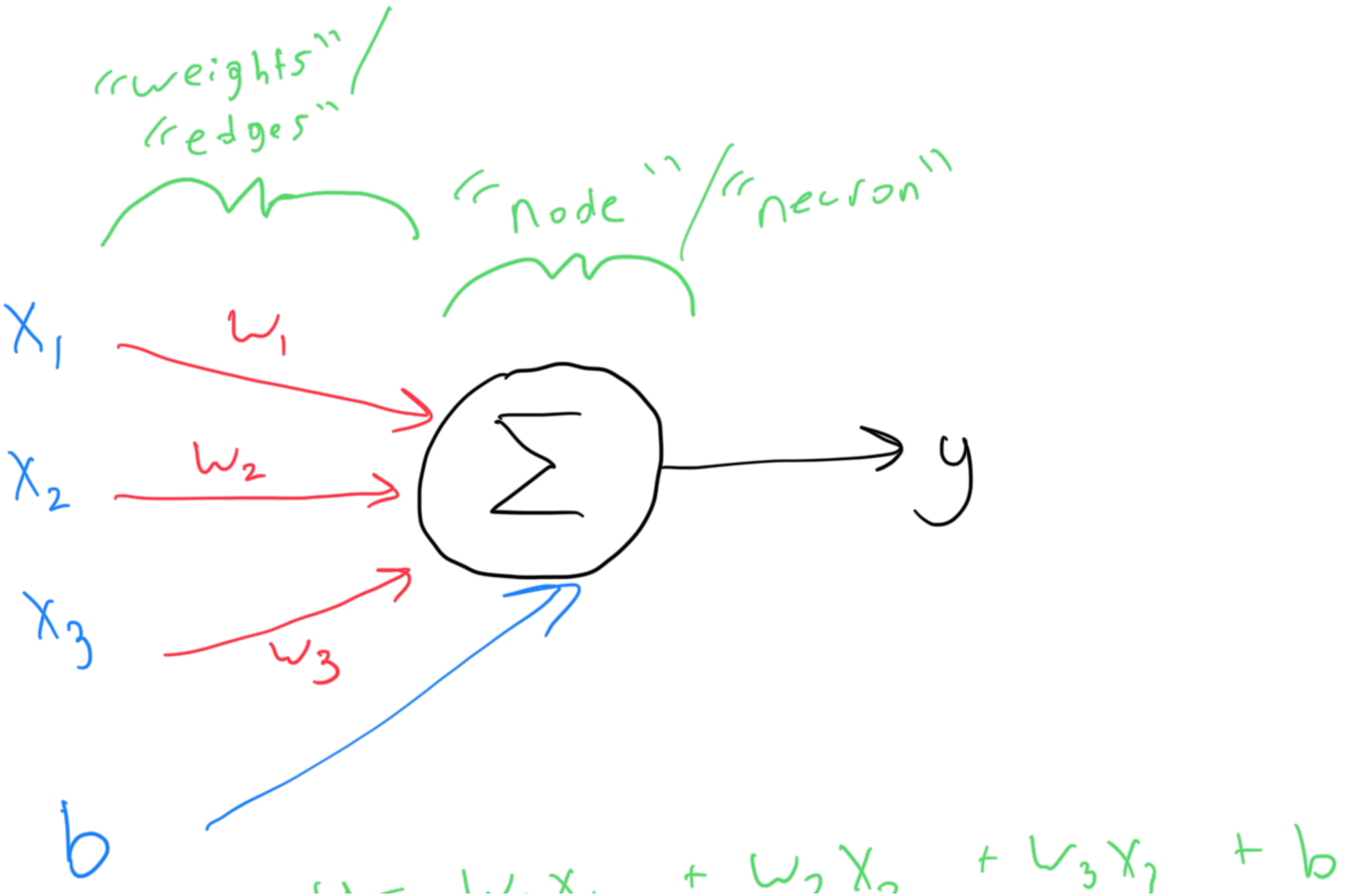
Day 20: Neural Network Fundamentals

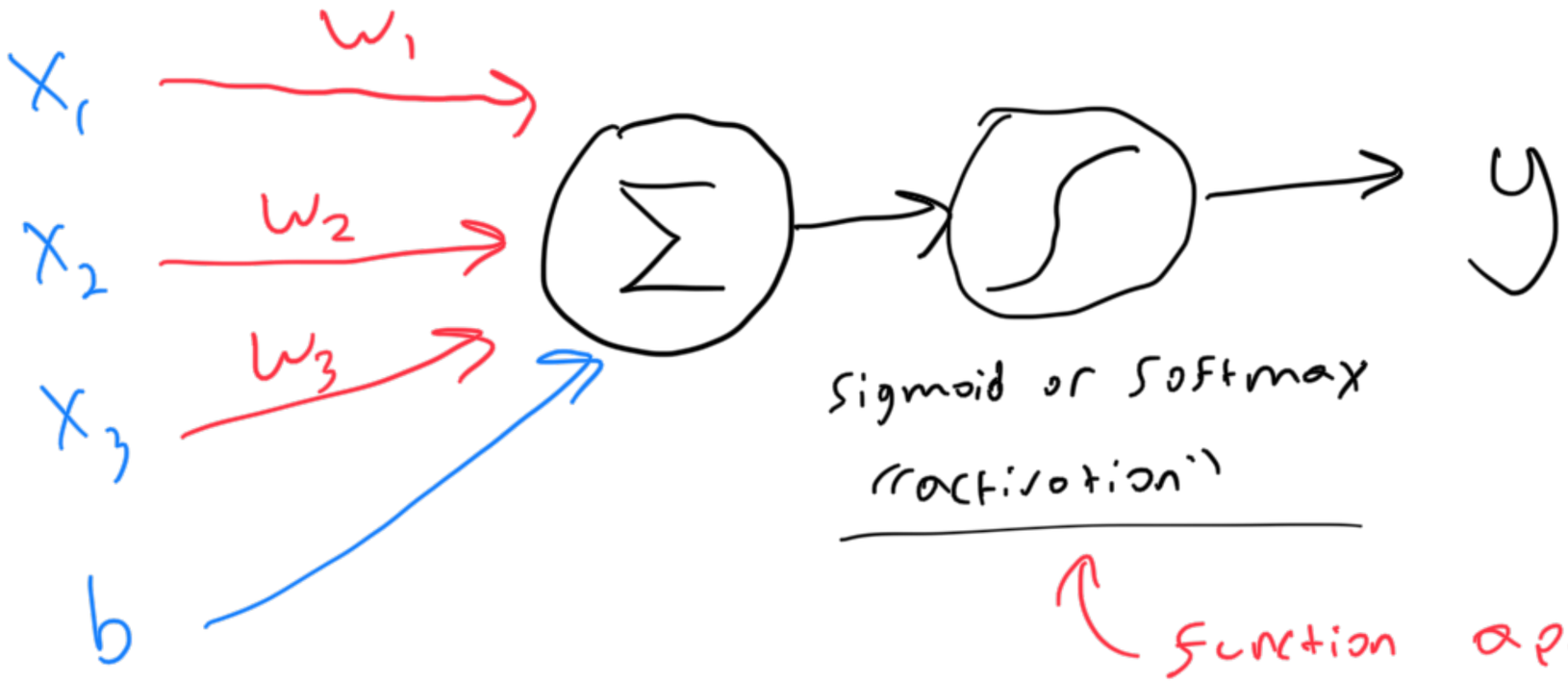
Recall: "Linear and Logistic Regression are neural networks"

Think of linear regression as:



$y = w_1 x_1 + \dots + z$

Think of logistic regression as:

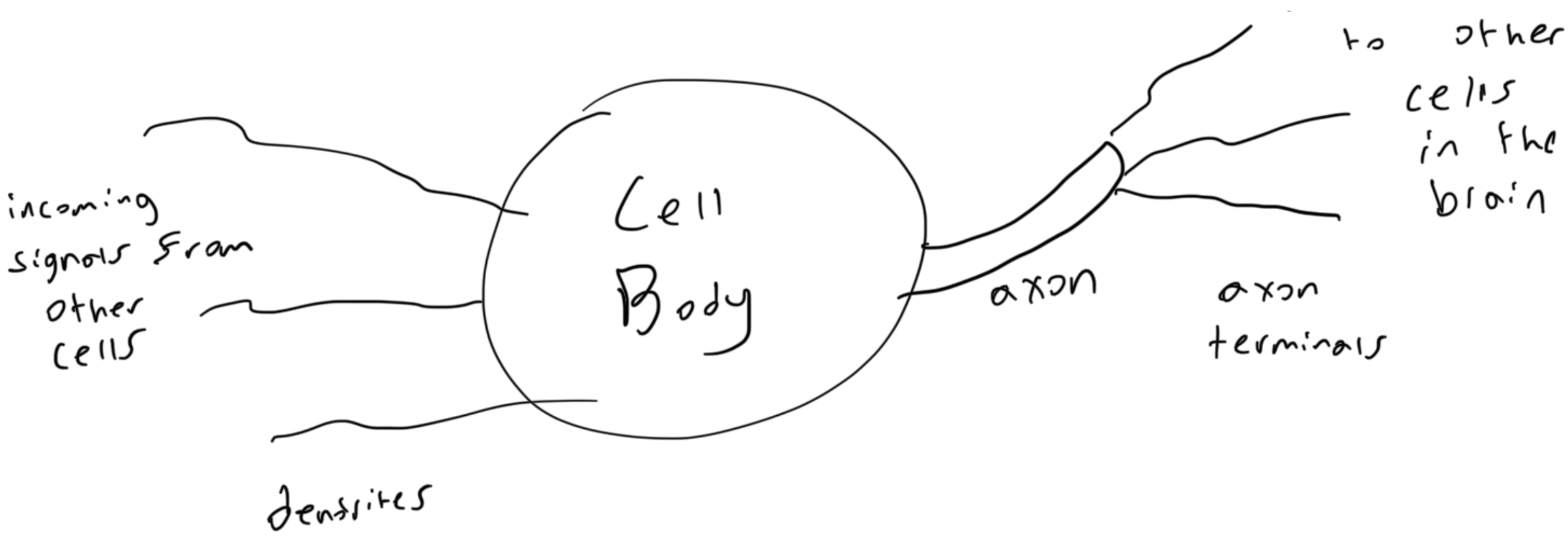


↑ function applied to the output of a node

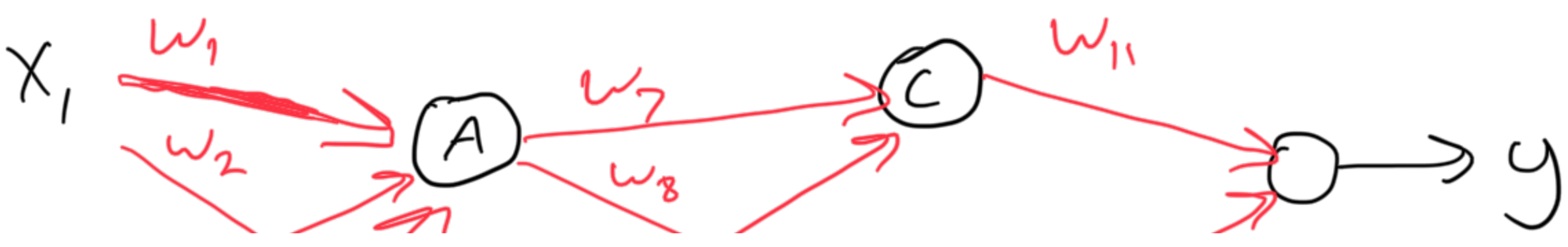
$$y = \text{sigmoid}\left(w_1 x_1 + w_2 x_2 + w_3 x_3 + b\right)$$

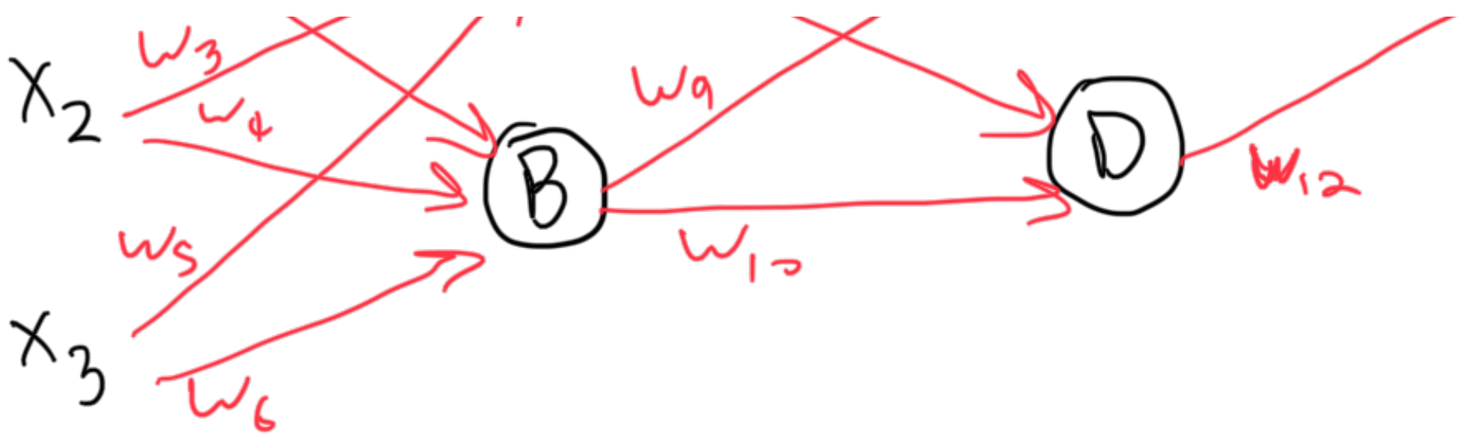
Above formulation is called a "Perceptron"

output signals



An artificial neural network is a set of connected layers of perceptrons, where the output of one perceptron is the input to the next perceptron





Input Layer

Hidden Layer

Hidden Layer

Output Layer

$$y = W_{11}C + W_{12}D$$

$$= W_{11}(W_7 A + W_9 B) + W_{12}(W_8 A + W_{10} B)$$

$$W_{11}\Big(W_7(W_1 X_1 + W_3 X_2 + W_5 X_3) + W_9(W_2 X_1 + W_4 X_2 + W_6 X_3)\Big)$$

$$+ W_{12}\Big($$

$$+ w_{12}\left(w_8\left(w_1 x_1 + w_3 x_2 + w_5 x_3\right)\right.$$
$$\left. + w_{10}\left(w_2 x_1 + w_4 x_2 + w_6 x_3\right)\right)$$

$$= w_{11} w_7 w_1 x_1 + w_{11} w_7 w_3 x_2$$
$$+ w_{11} w_7 w_5 x_3 + w_{11} w_9 w_2 x_1 + \cdots$$

Isn't this a linear function?

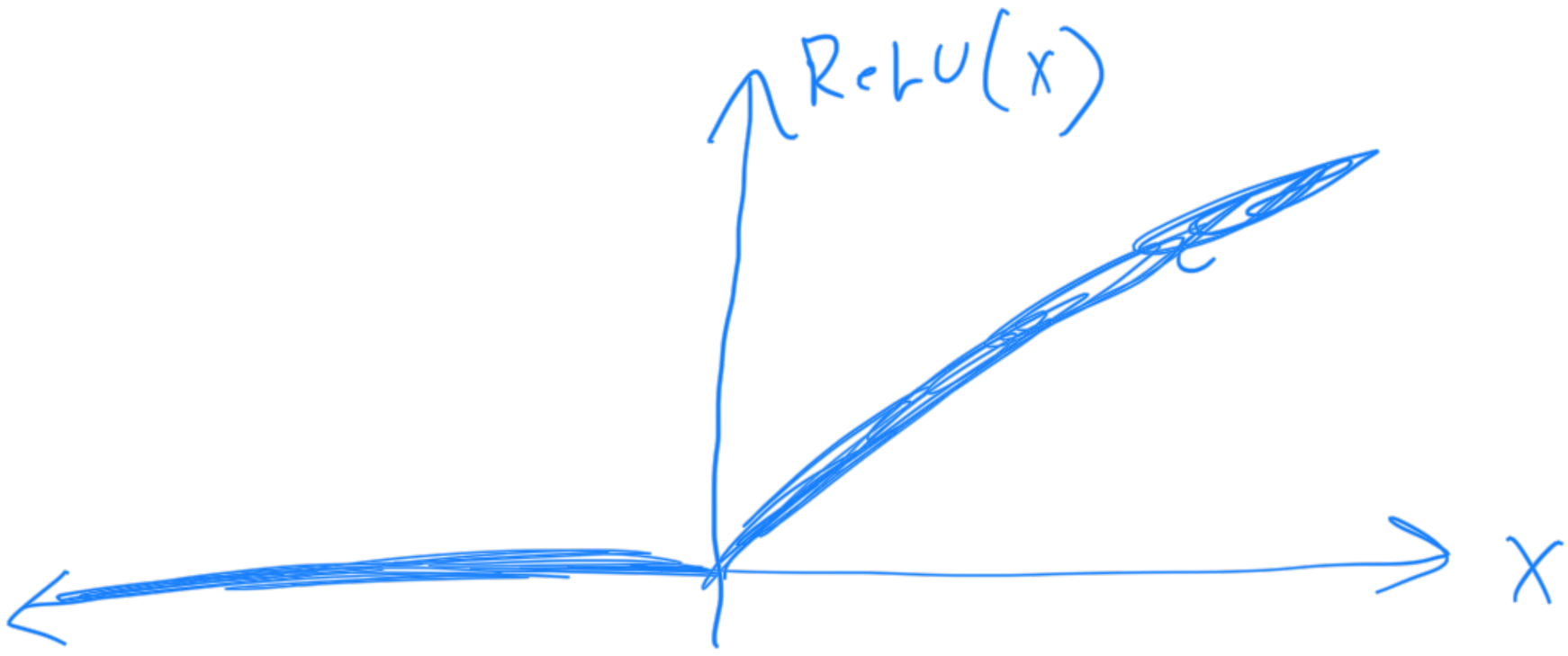
Why not use linear/logistic regression?

Answer: add non-linearity to the network!!!

Take each node's output, and apply a non-linear activation function to it.

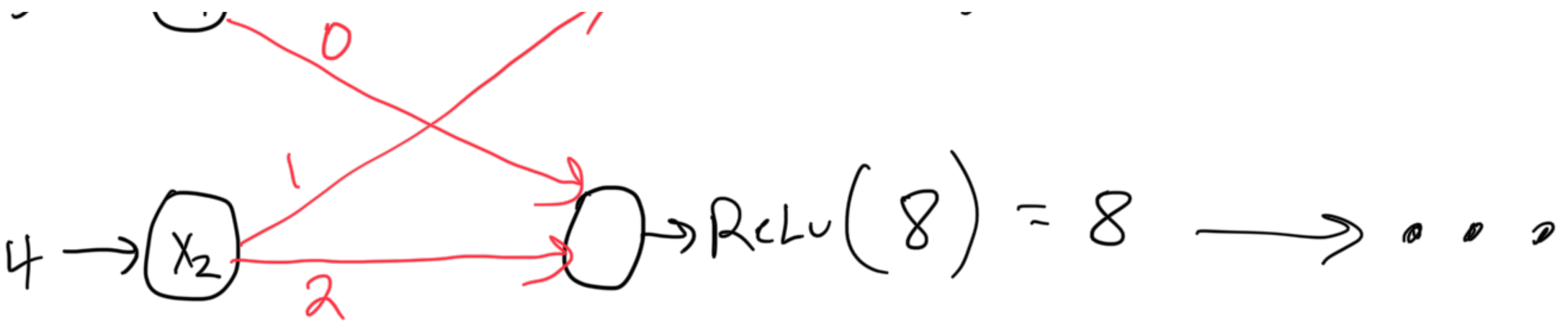
Popular activation function: ReLU

"Rectified Linear Unit"



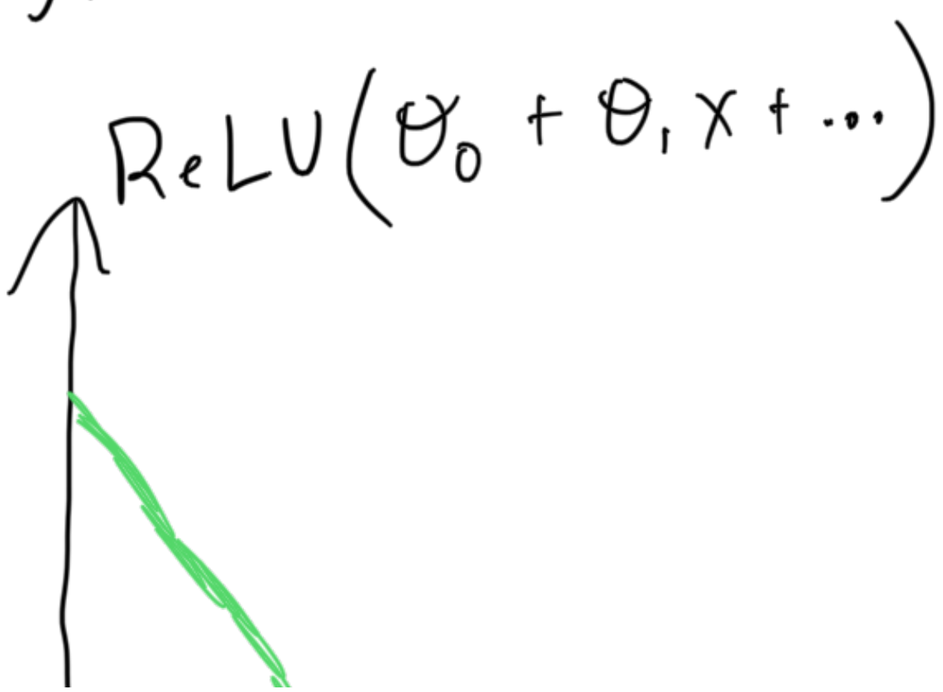
$$ReLU(x) = \max(0, x)$$

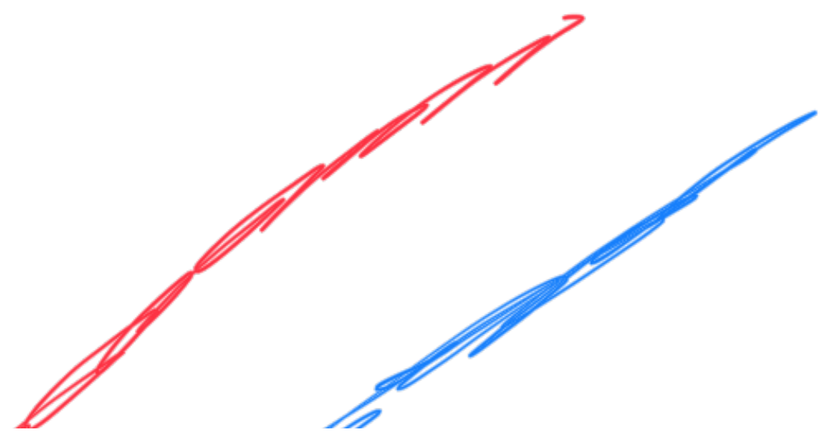
$$= \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

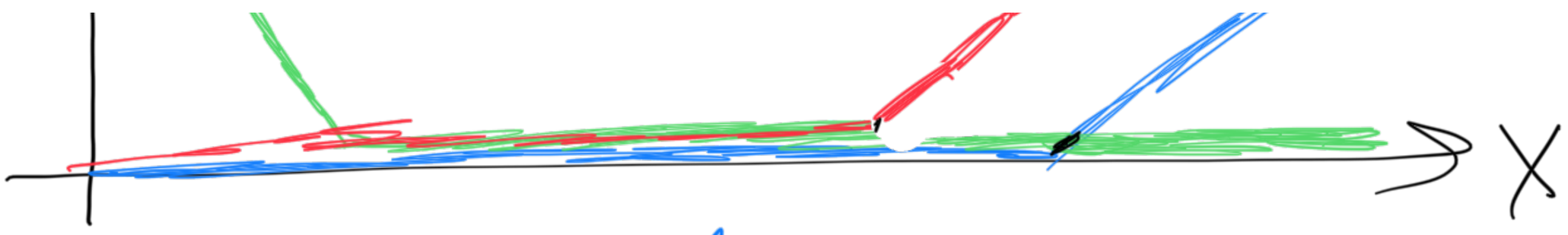


Activation functions like ReLU allow networks to learn ANY function (given enough nodes and edges)

Why?





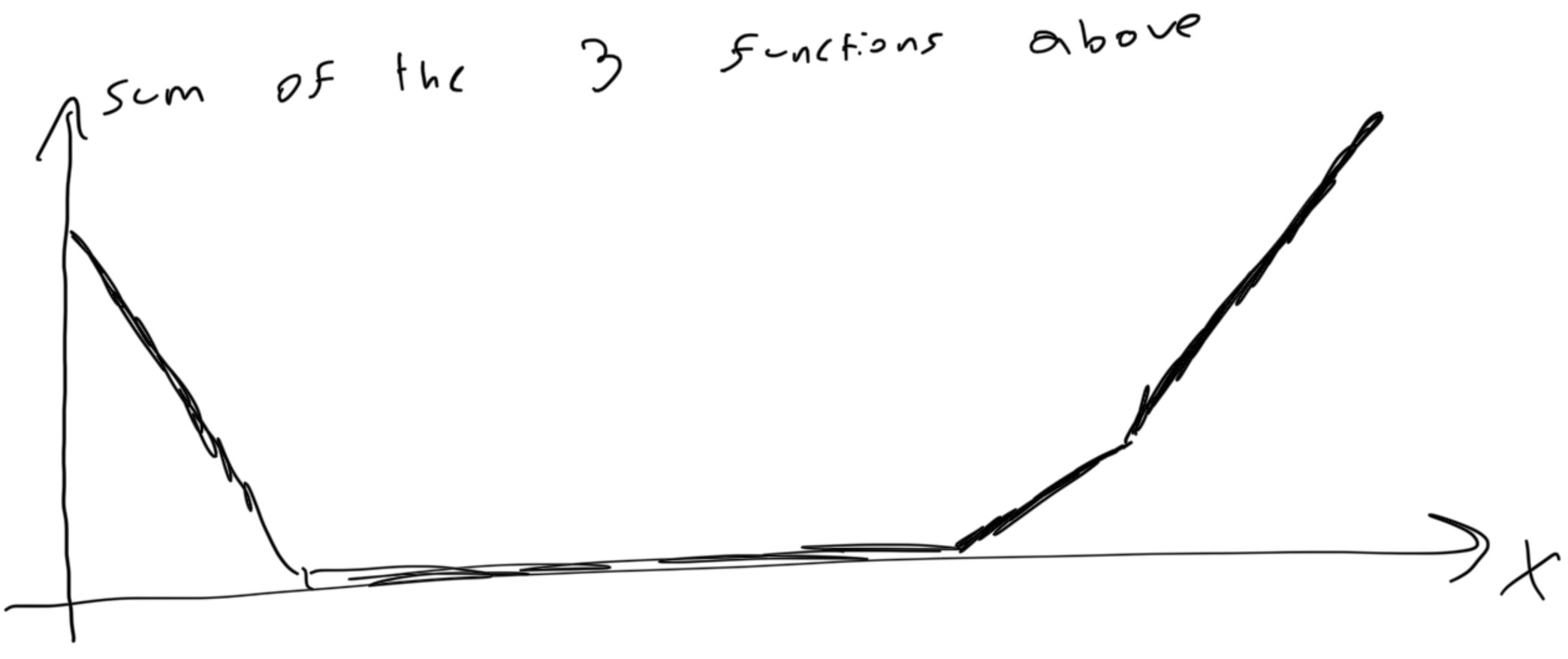


= 0 set 1

= 0 set 2

= 0 set 3

When adding these up, we get:



For more functions,

If you [illegible]
you can approximate anything!

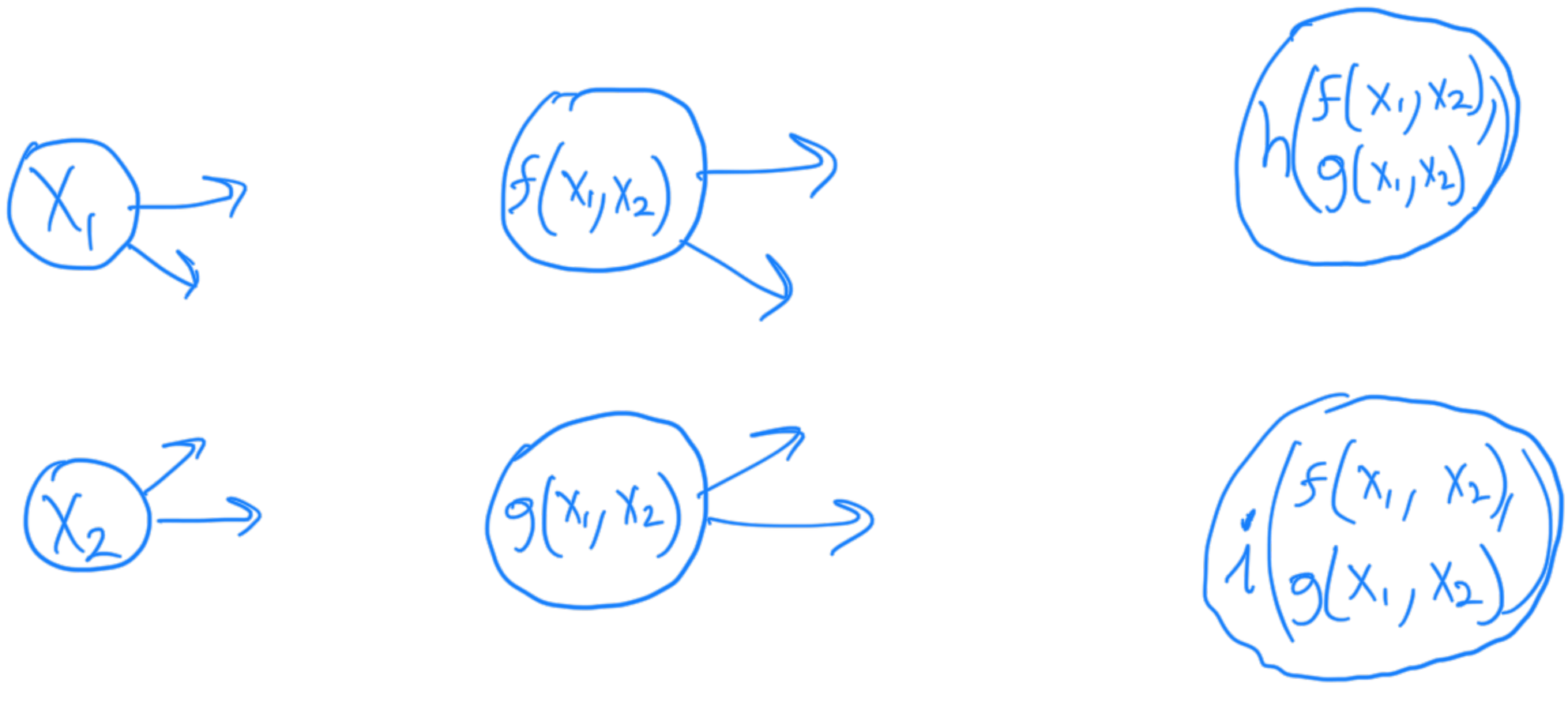
Universal Approximation Theorem

"Deep Learning" = Neural net (NN) w/ many layers

Why use deep learning (DL) when we can approximate anything with 1 hidden layer?

Deep networks need exponentially fewer Θ

to achieve similar performance



Stacking layers allows using prior layers as modular functions

→ can create more complex functions w/ same # of weights

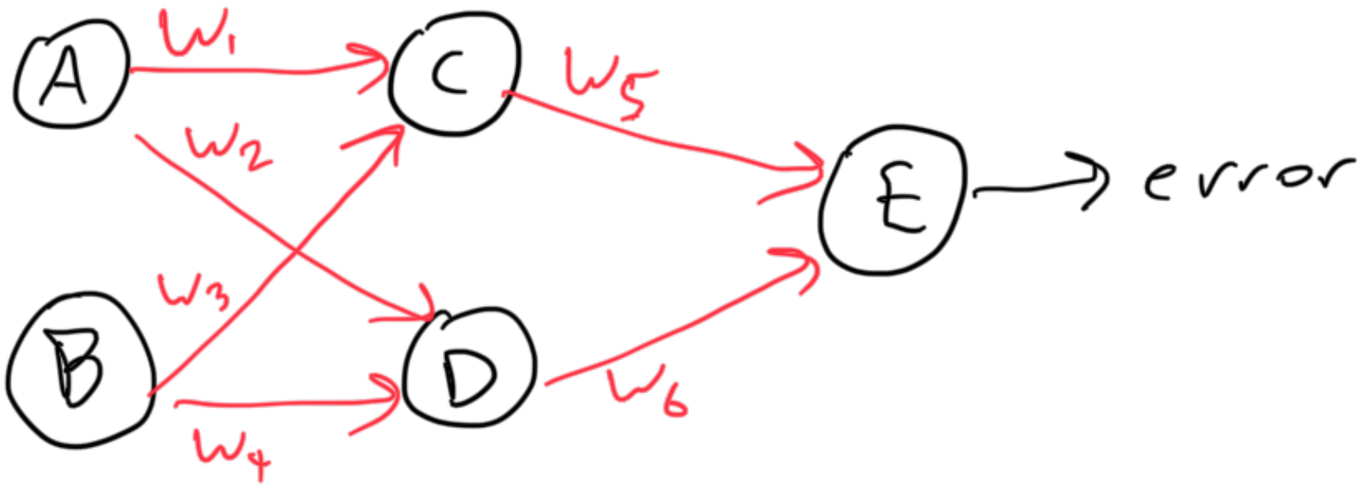
Progressive layers learn more high-level/abstract concepts

How are NN trained?

Gradient Descent, like before

$$W_i = W_i - \alpha \frac{\partial \text{Loss}}{\partial W_i}$$

But weights are layered;



$\frac{\partial \text{Loss}}{\partial W_1}$ depends on W_5, C, and E

Use the Chain Rule

If $y = f(a)$ where $u = g(x)$

Then:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Backpropagation

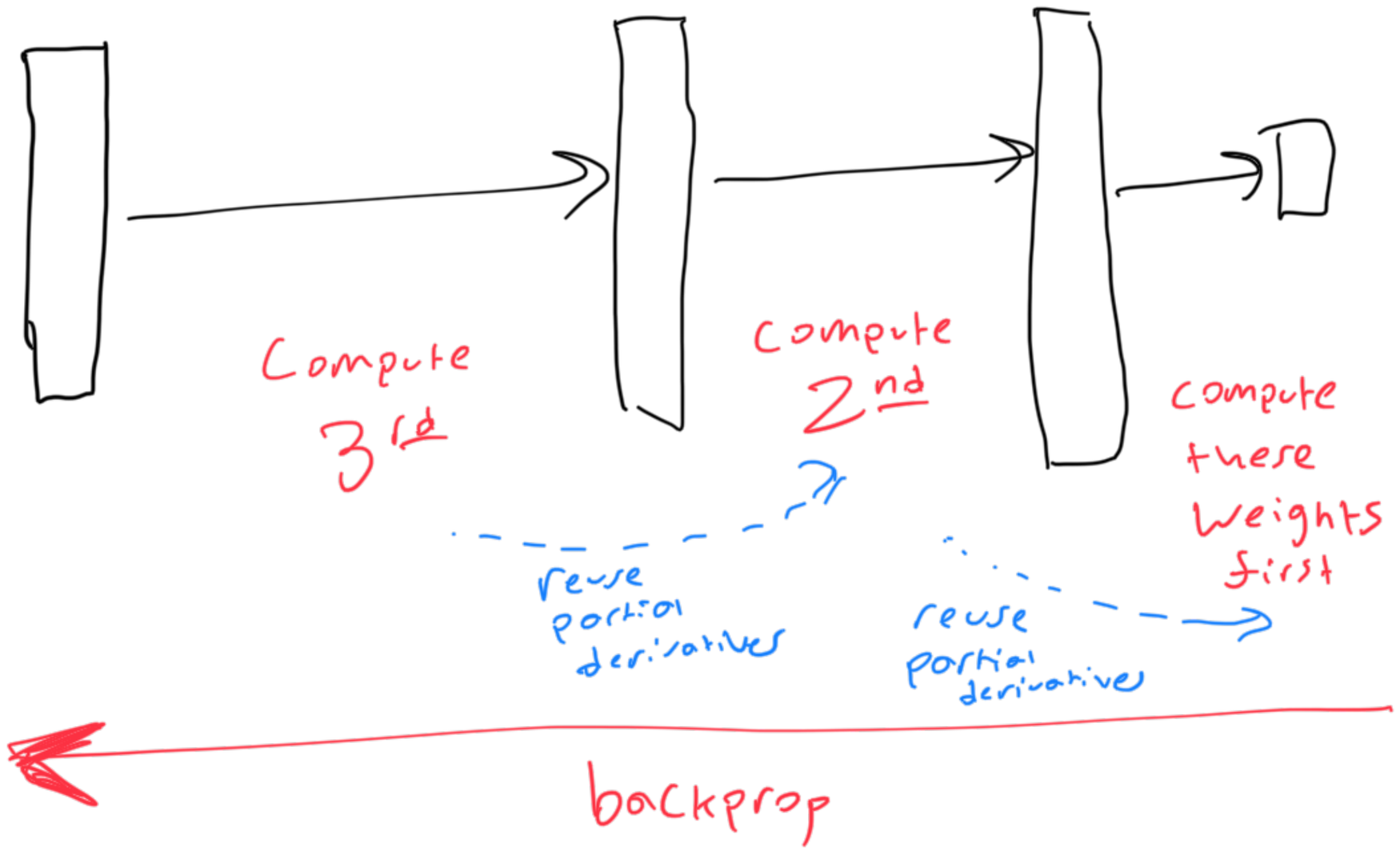
Backprop Example

$$\frac{\partial Loss}{\partial w_{31}} = \underbrace{\frac{\partial Loss}{\partial \sigma_3} \cdot \frac{\partial \sigma_3}{\partial z_3}}_{} \cdot \frac{\partial z_3}{\partial w_{31}}$$

$$\frac{\partial Loss}{\partial w_{21}} = \underbrace{\frac{\partial Loss}{\partial \sigma_3} \cdot \frac{\partial \sigma_3}{\partial z_3}}_{\text{already computed in}} \cdot \frac{\partial z_3}{\partial \sigma_1} \cdot \frac{\partial \sigma_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{21}}$$

the previous step

We can use dynamic programming to cache results of prior computations



"Multi-Layer Perceptron (MLP)

or

"Dense Neural Network"

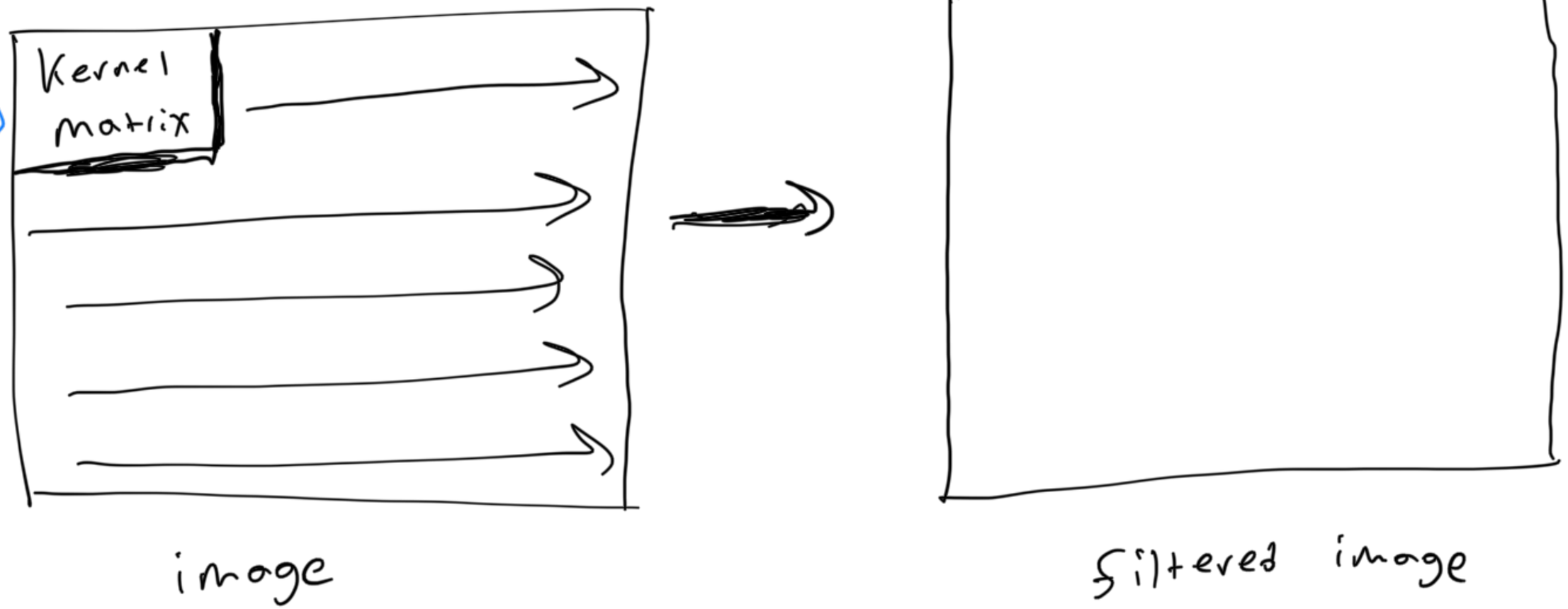
are the "basic" type of NN we have learned about so far

Convolutional Neural Networks (CNNs)

Short Review

extract visual properties of

Recall: We can [illegible] on image through convolutions:



image

filtered image

Recall: Kernels are specialized to filter images differently.

-1	-1	-1
-1	8	-1

outline kernel

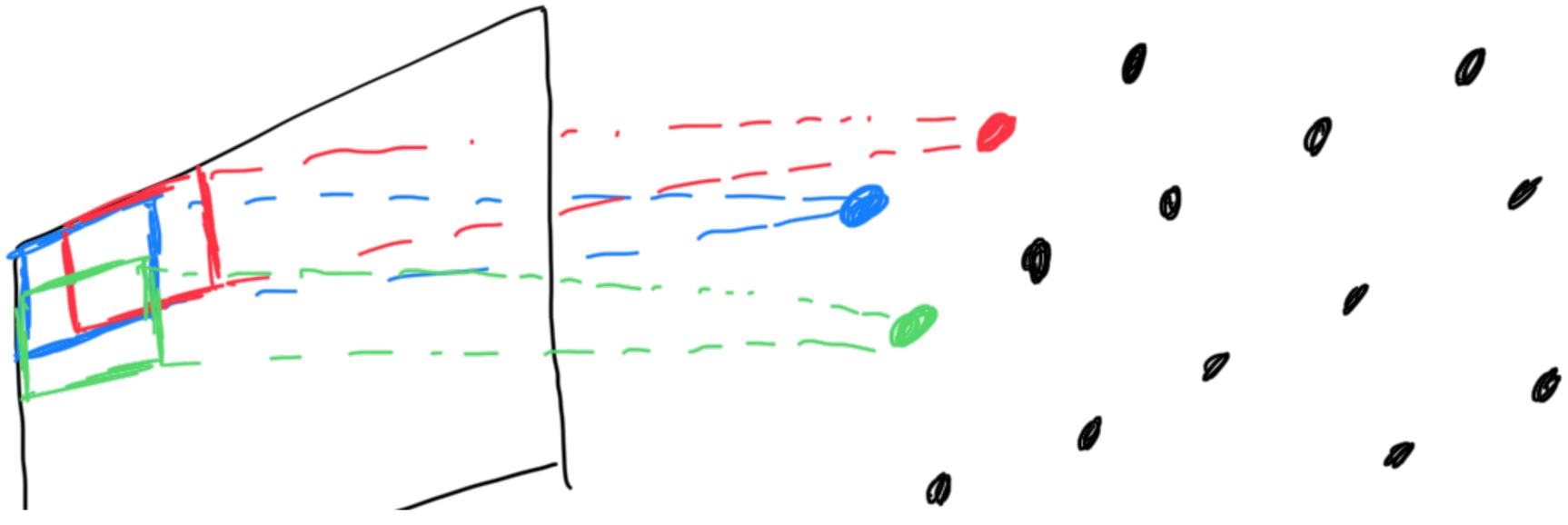
-1	-1	-1

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

blur kernel

0	-1	0
-1	5	-1
0	-1	0

Sharper Kernel

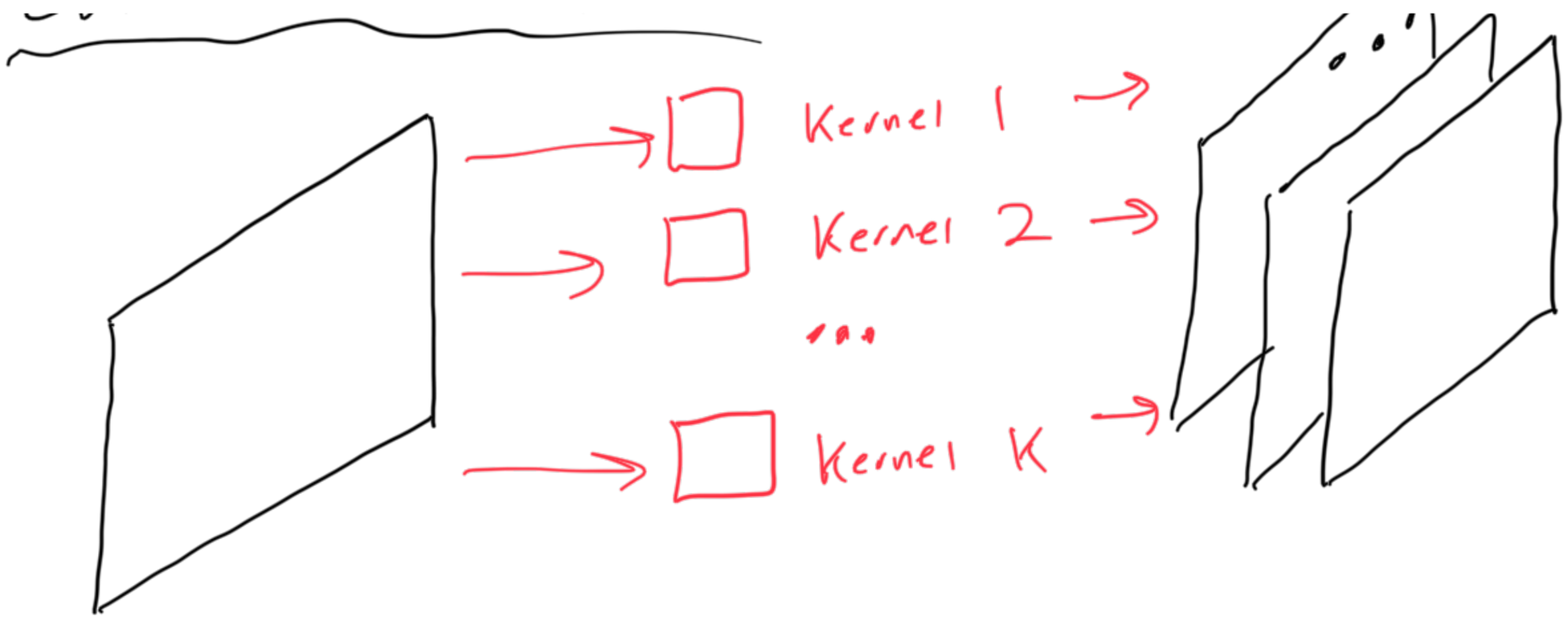


each pixel in new image is dot product between the kernel and corresponding image portion

Key ideas of CNNs:

1. use several kernels
2. learn the best kernels to use (i.e., best features to extract)

Convolutional Layers


Kernel 1
Kernel 2
Kernel K