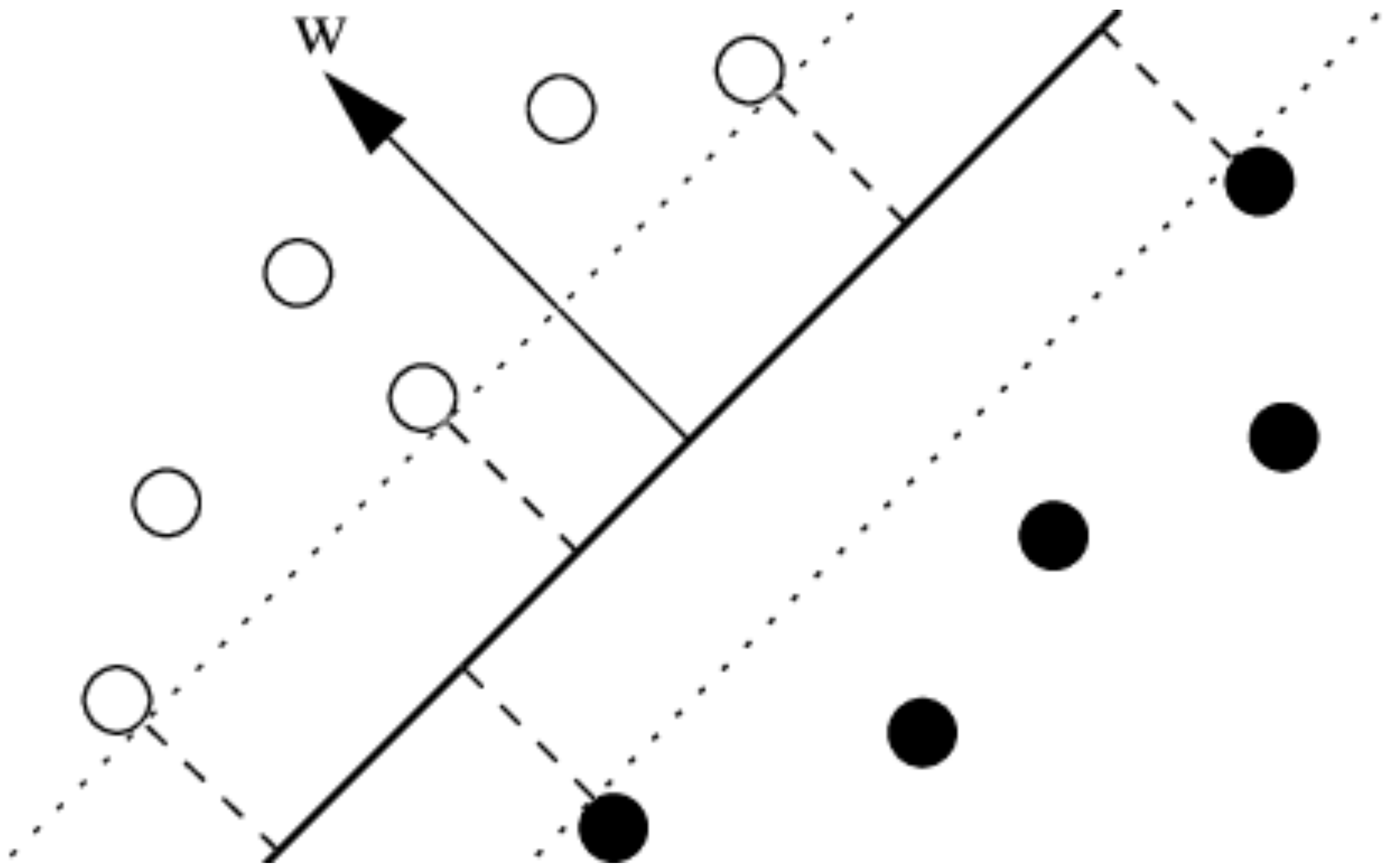
13. Deep Learning

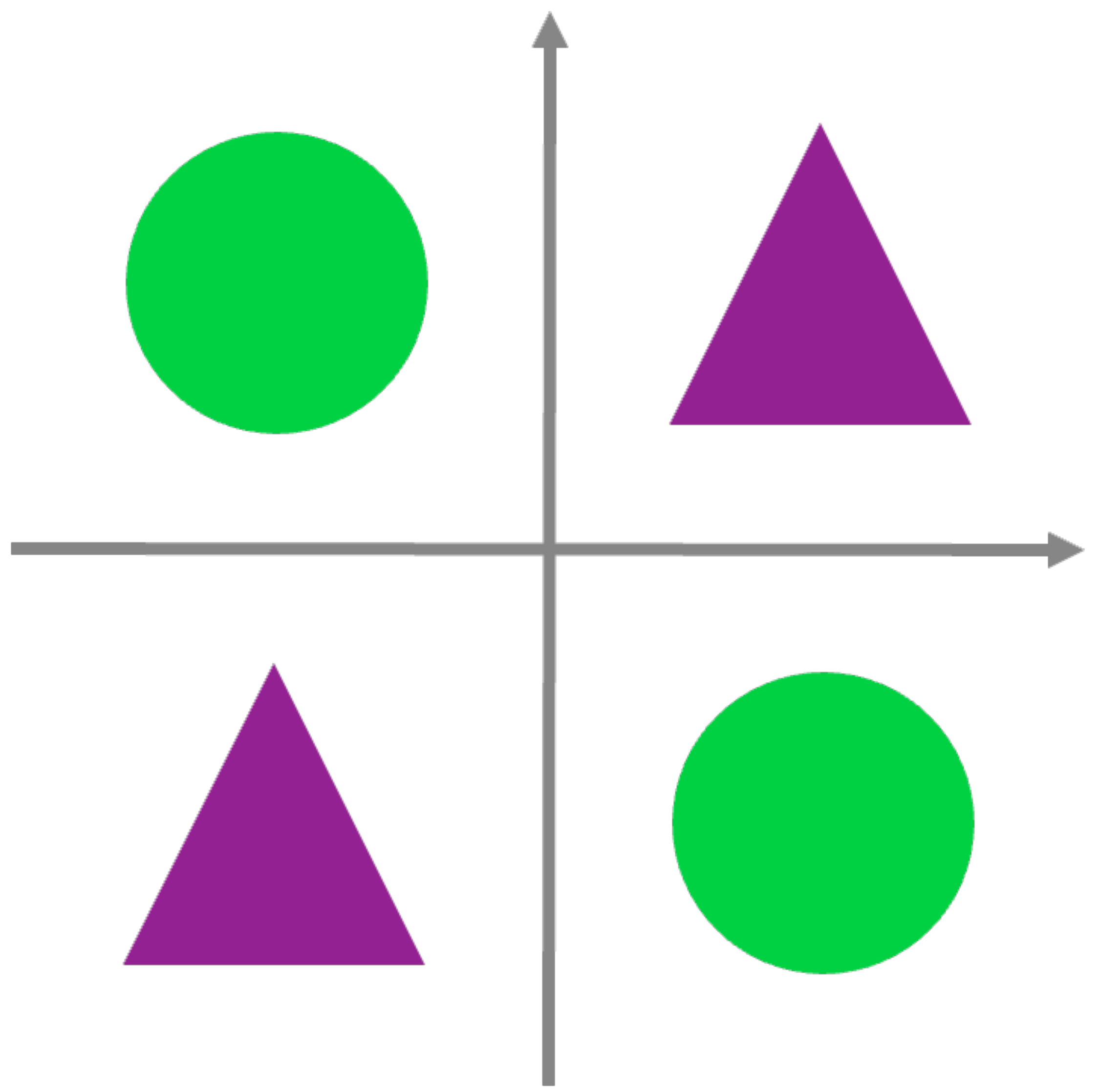
EEC454 Introduction to Machine Learning Systems

2023 Fall, Jaeho Lee

Recap: Linear Models

- We have studied many **linear models**: perceptrons, SVM, ...
  - Easy to fit, but had limited **expressive power**.
    - Cannot perfectly predict on training data

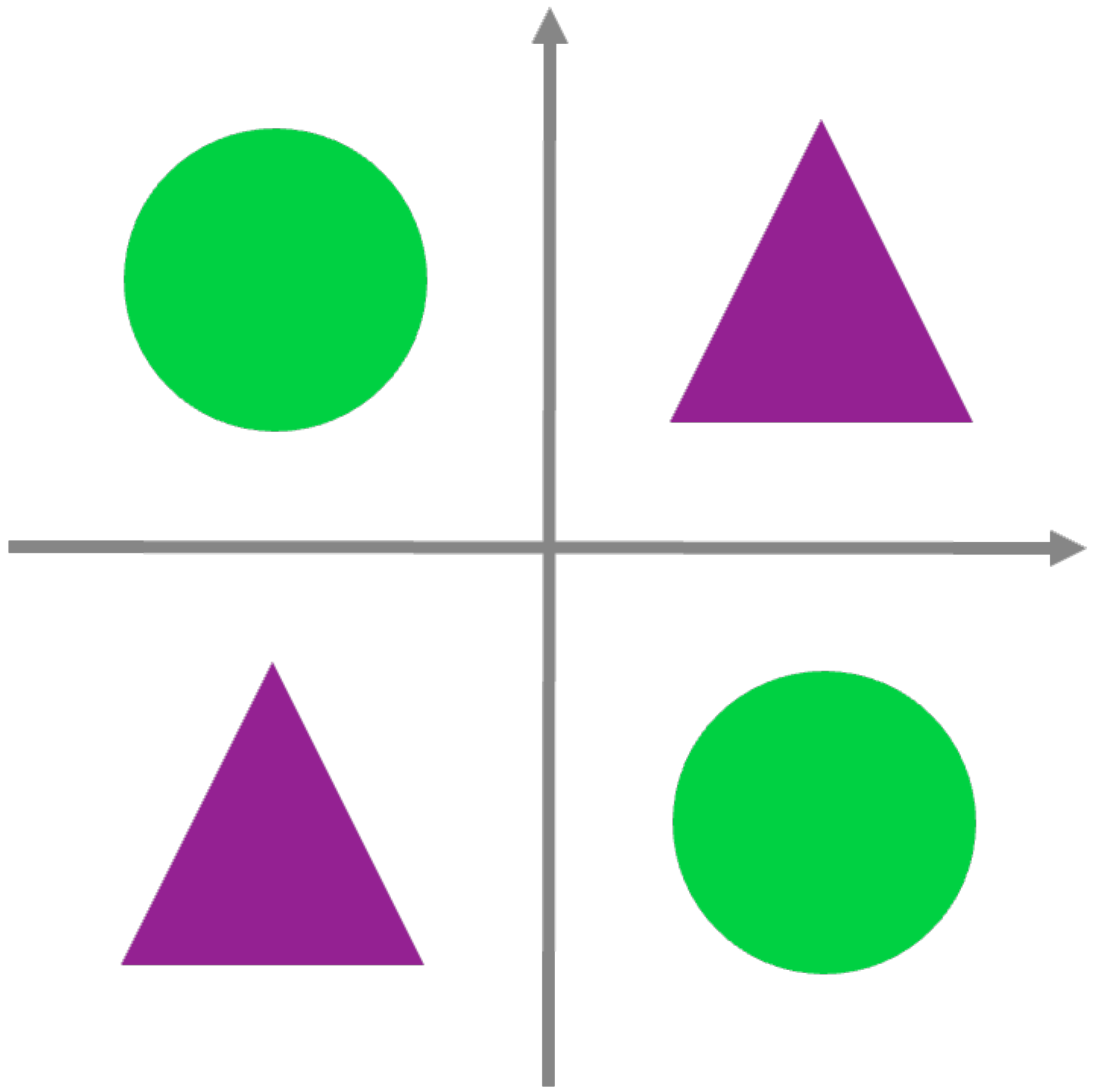




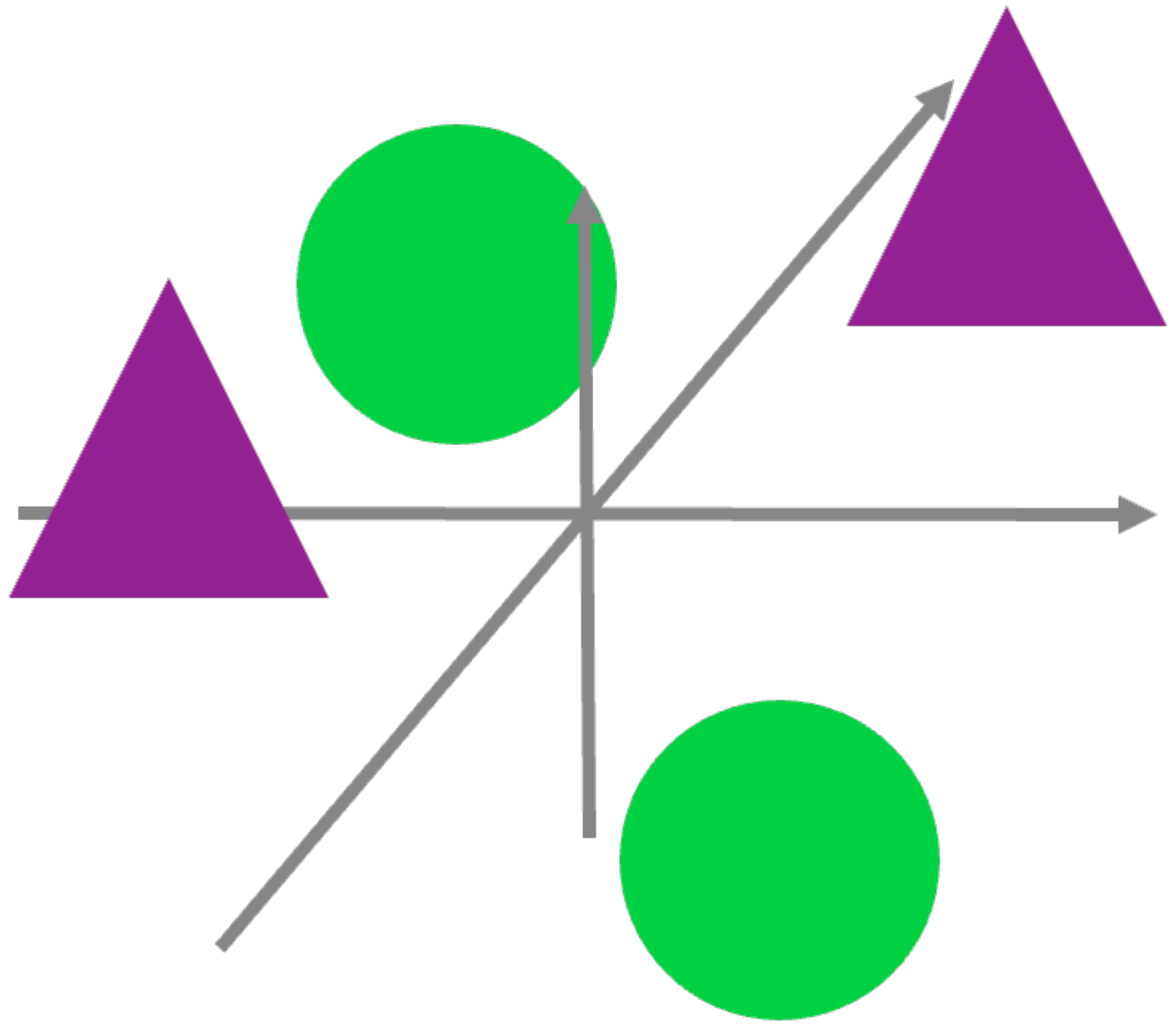
(x_1, x_2)

Recap: Feature maps

- A useful approach is to use the **feature map**.
  - A good linear model may exist in higher-dimensional space.
    - We used **handcrafted features**, usually...



(x_1, x_2)



$(x_1, x_2, x_1 x_2)$

Features (a.k.a. Representations)

- Mathematically put, we were solving:

$$\min_{\Phi(\cdot)} \min_{\text{linear } f(\cdot)} \frac{1}{n}\sum_{i=1}^{n} \ell\Big(y_i, f(\Phi(\mathbf{x}_i))\Big)$$

Human trial-and-error...?

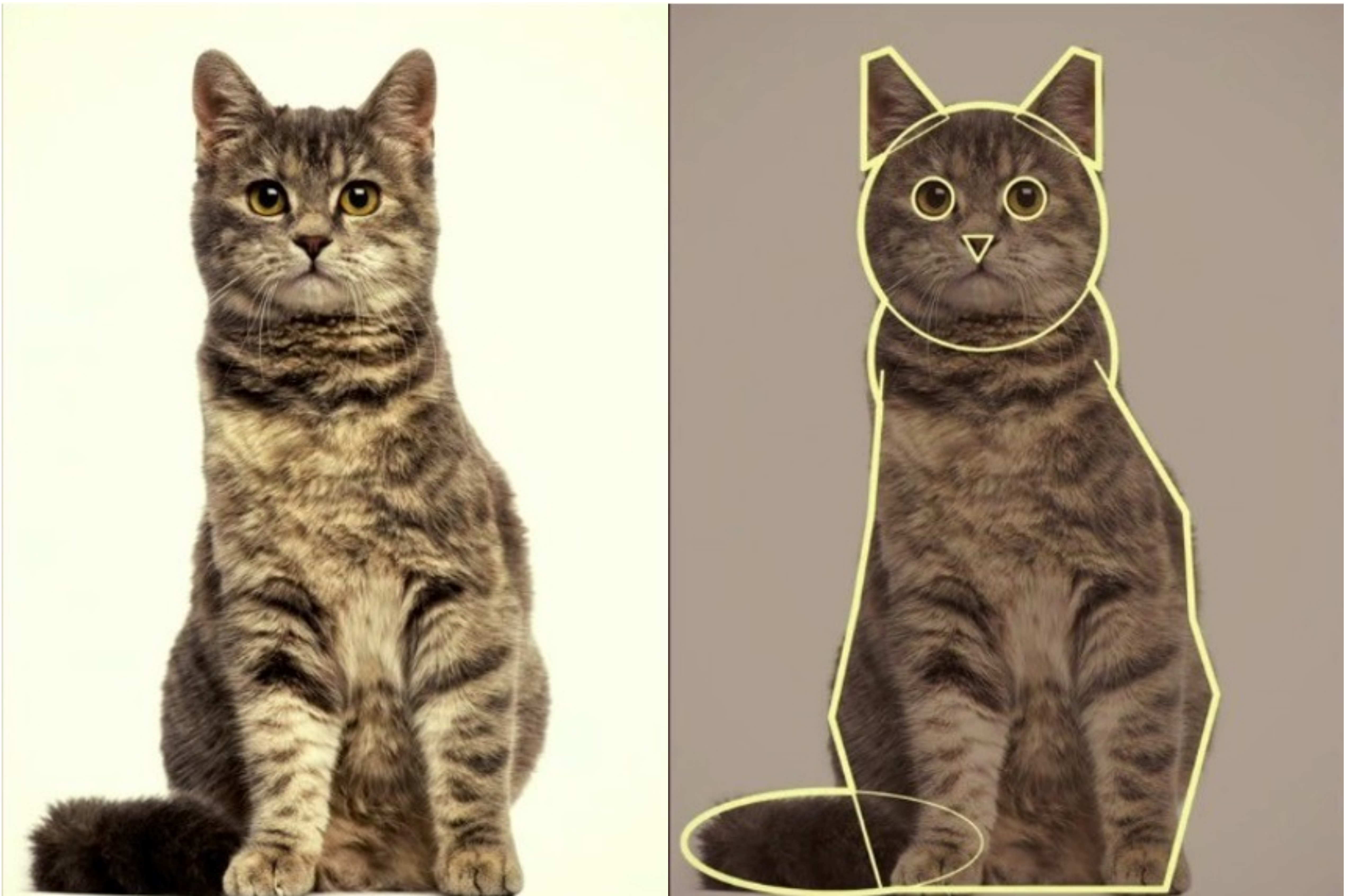
Automated optimization, with data

- **Problem.** Crafting a nice $\Phi(\cdot)$ for complicated data is quite difficult...

Image source: CS231n @ Stanford

Features (a.k.a. Representations)

- Consider a cat detector.
  - We may use some domain knowledge to build good features.



$\phi_1(\mathbf{x})$ = "round head"

$\phi_2(\mathbf{x})$ = "two triangular ears"

$\phi_3(\mathbf{x})$ = "two round eyes"

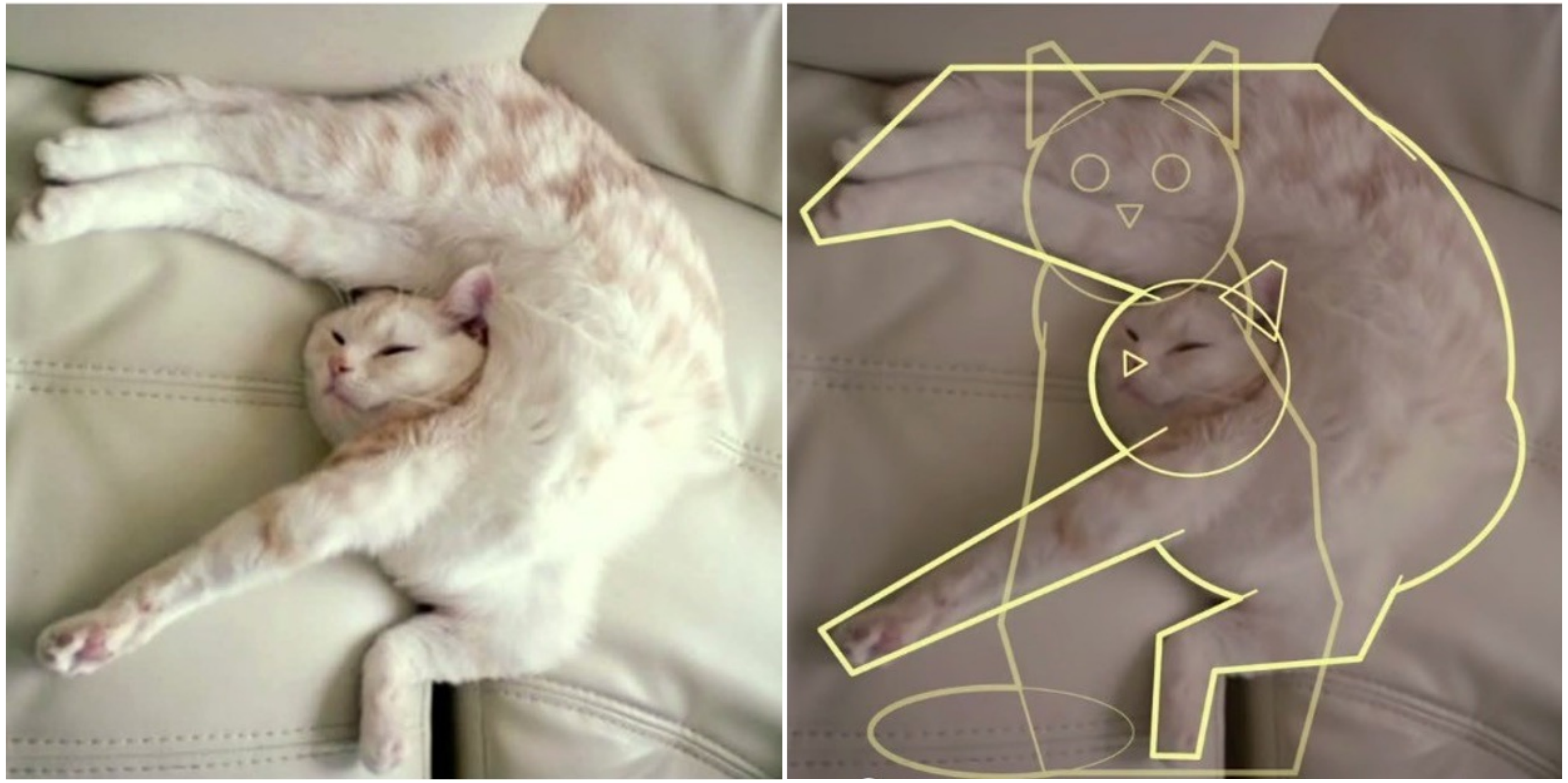
$\phi_4(\mathbf{x})$ = "oval tail"

$\phi_5(\mathbf{x})$ = …

Image source: CS231n @ Stanford

Features (a.k.a. Representations)

- Consider a cat detector.
    - We may use some domain knowledge to build good features.



$\phi_1(\mathbf{x})$ = "round head" O

$\phi_2(\mathbf{x})$ = "two triangular ears" X

$\phi_3(\mathbf{x})$ = "two round eyes" X

$\phi_4(\mathbf{x})$ = "oval tail" X

$\phi_5(\mathbf{x})$ = ...

Image source: CS231n @ Stanford

Features (a.k.a. Representations)

- Consider a cat detector.
  - We **may not** use domain knowledge to build good features.

Representation Learning

$$\min_{\Phi(\cdot)} \min_{\text{linear } f(\cdot)} \frac{1}{n} \sum_{i=1}^{n} \ell\Big(y_i, f(\Phi(\mathbf{x}_i))\Big)$$

Automated optimization, with data

- **Representation learning** learns $\Phi(\cdot)$ from data.
  - jointly optimized with f (typically when there's many labeled data)
  - separately obtained from unlabeled data
  - both

Deep Learning

- **Q1.** How do we parameterize $\Phi(\cdot)$? (i.e., hypothesis space for Φ)
  - **Desired.** Rich enough, so that it can express complicated functions
    - *Deep neural networks*

- **Q2.** How do we optimize such $\Phi(\cdot)$?
    - *Gradient descent, using backpropagation*

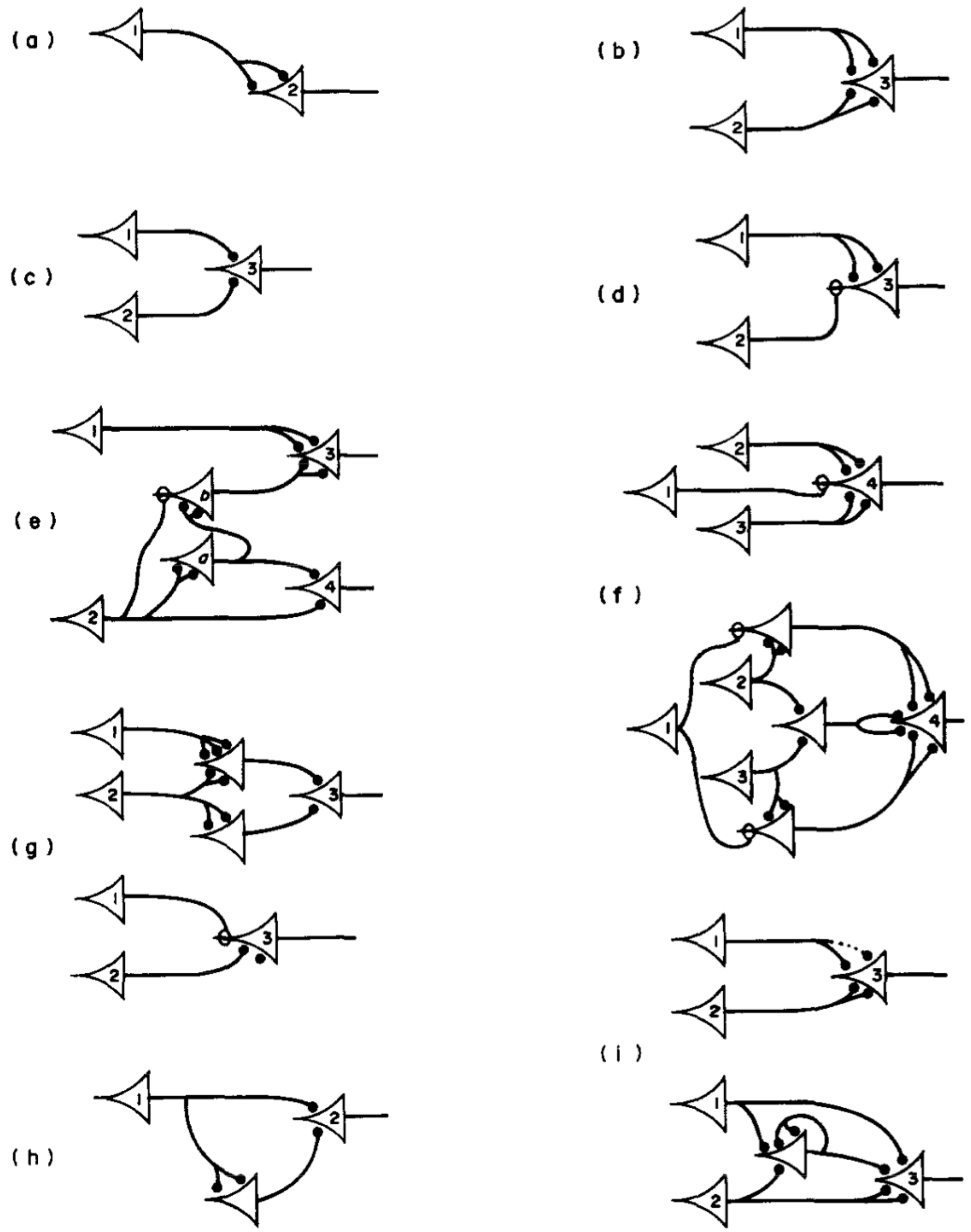
(Deep) Neural Networks

Neural Network

- Inspired by how human processes info.
  (now very far from the human biological details)

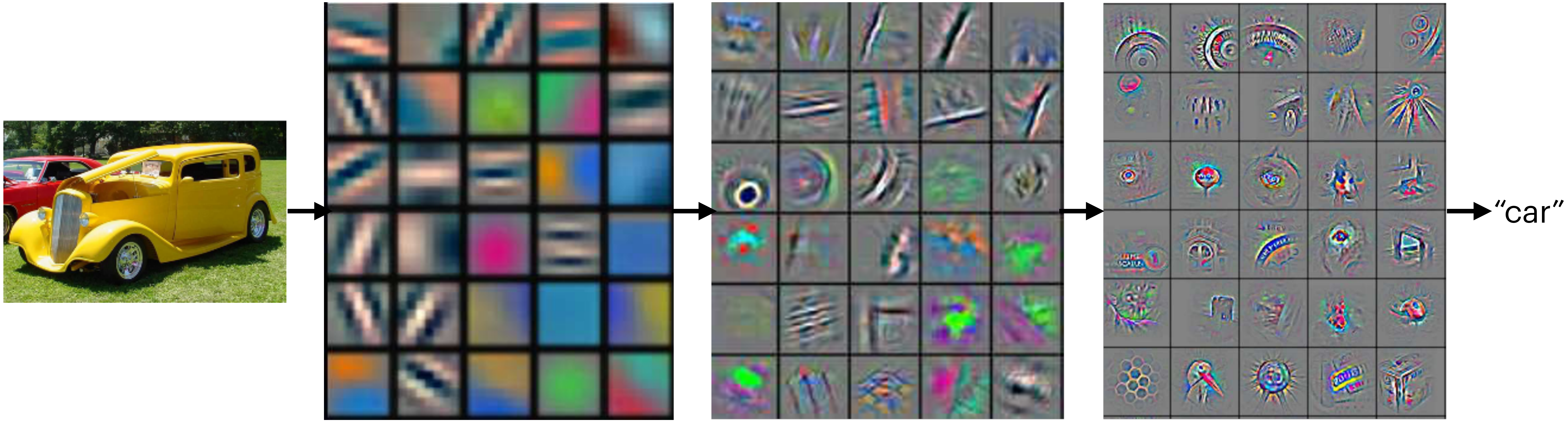
A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY*

■ WARREN S. MCCULLOCH AND WALTER PITTS
University of Illinois, College of Medicine,
Department of Psychiatry at the Illinois Neuropsychiatric Institute,
University of Chicago, Chicago, U.S.A.



Neural Network

- **Idea.** Human processes information using **multiple layers of neurons**.
  - Each individual neuron performs a **simple operation**.
  - Neurons **sequentially build** more complicated information.



Multi-Layer Perceptrons (MLPs)

- Recall that **perceptrons** use the classifier of form

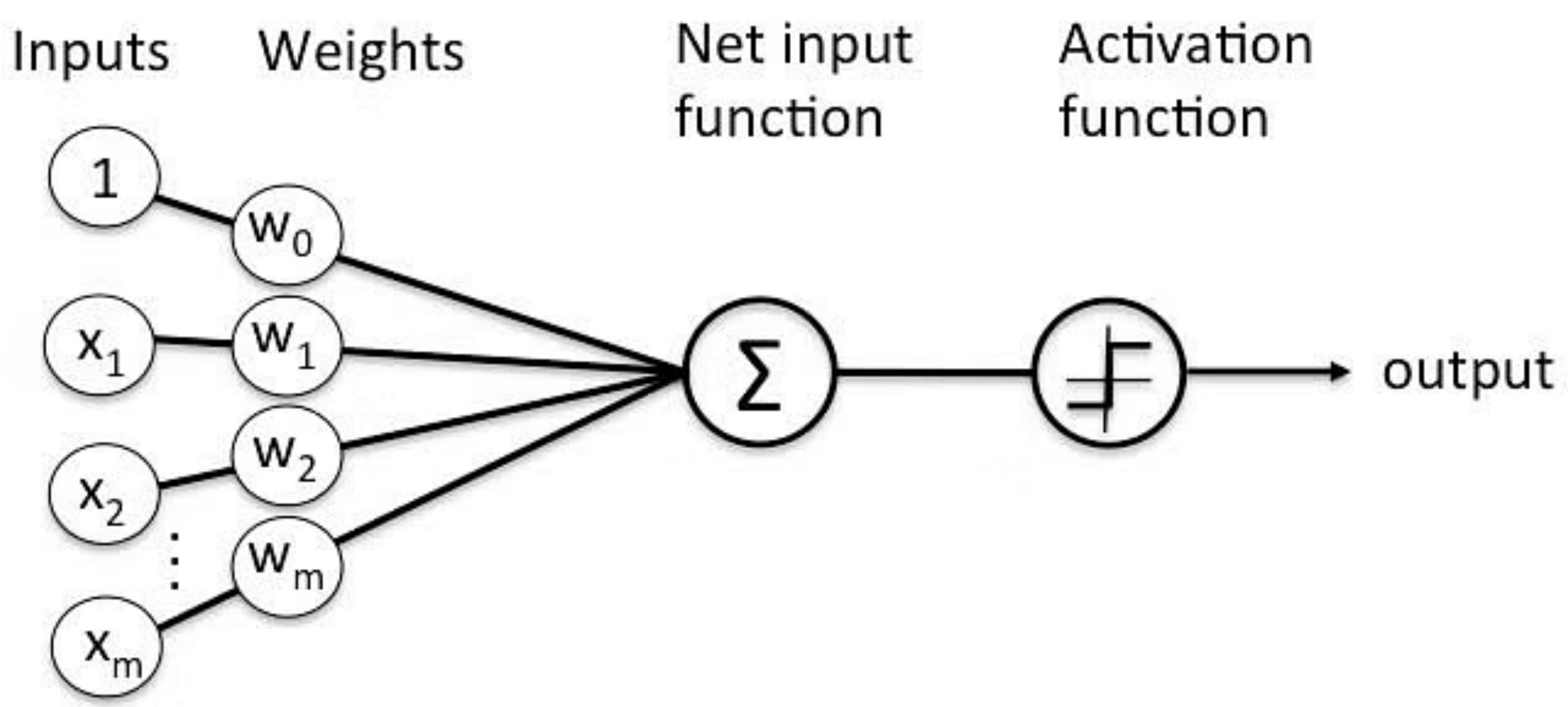
$$f_\theta(\mathbf{x}) = \mathbf{1}[\theta^\top \mathbf{x} > 0]$$

- This is a combination of two functions:
  - A linear operation

$$\mathbf{x} \mapsto \theta^\top \mathbf{x}$$

  - A nonlinearity (or activation function)

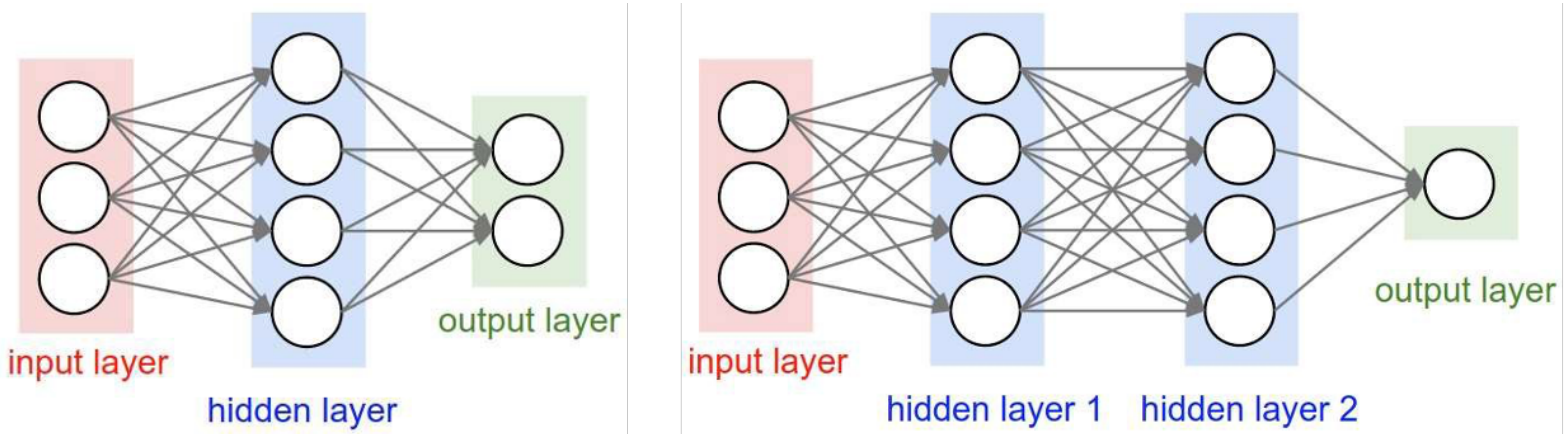
$$\mathbf{x} \mapsto \mathbf{1}[\mathbf{x} > 0]$$



Multi-Layer Perceptrons (MLPs)

- **Multi-layer perceptrons** are a cascade of multiple parallel perceptrons.
- In the i-th layer, we do
  - Linear operation $\mathbf{z} \mapsto \mathbf{W}_i\mathbf{z} + \mathbf{b}_i$ (weights: $\mathbf{W}_i$; biases: $\mathbf{b}_i$)
  - Activation function $\mathbf{z} \mapsto \sigma_i(\mathbf{z})$ (typically applied entrywise)

  hidden layer activation; internal representation; ...



Multi-Layer Perceptrons (MLPs)

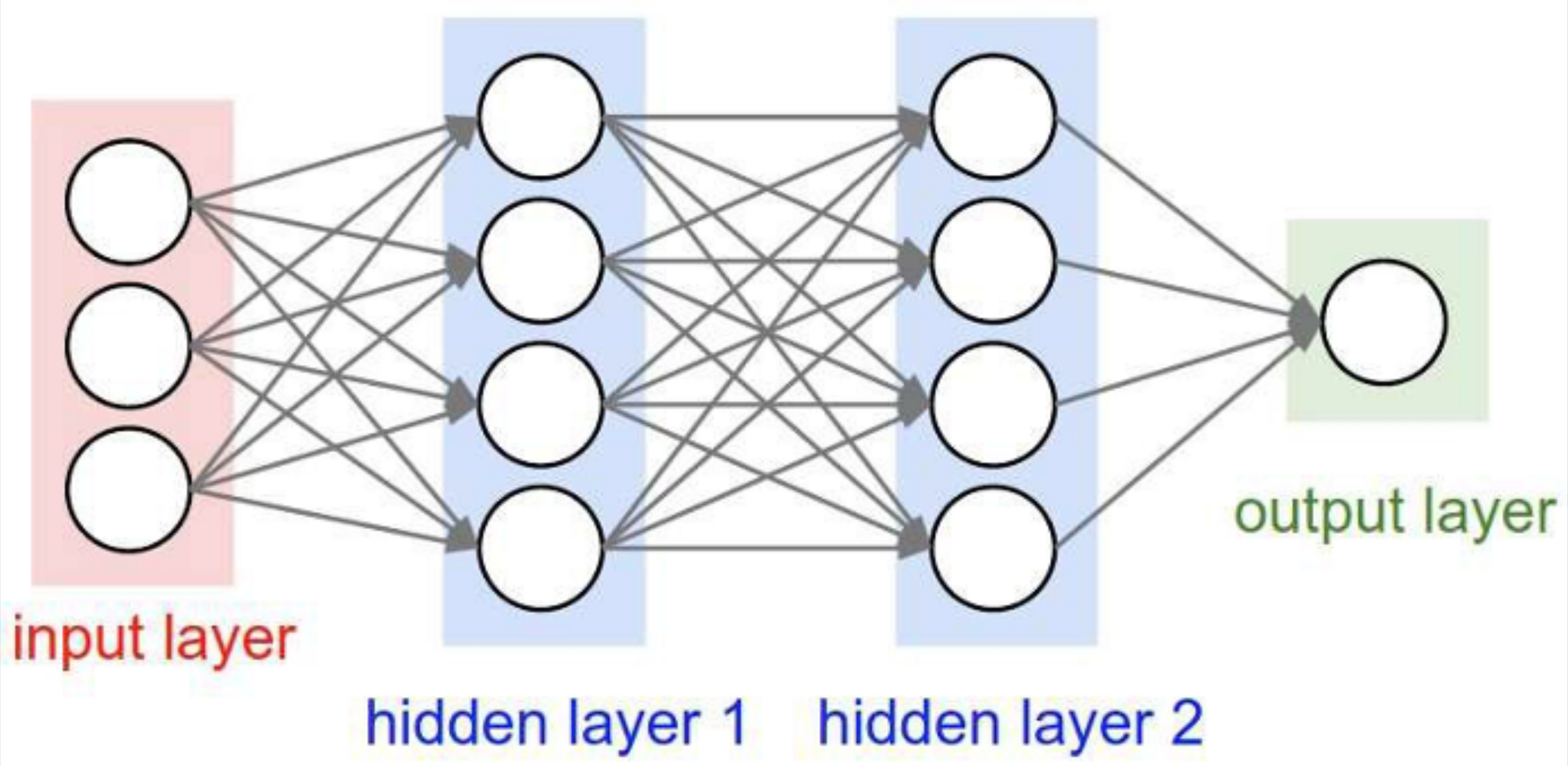
- **Multi-layer perceptrons** are a cascade of multiple parallel perceptrons.
- In the i-th layer, we do
  - Linear operation $\mathbf{z} \mapsto \mathbf{W}_i\mathbf{z} + \mathbf{b}_i$
  - Activation function $\mathbf{z} \mapsto \sigma_i(\mathbf{z})$ (typically applied entrywise)

- Ignoring the bias terms $\mathbf{b}_i$, our predictor can be written as:

$$f(\mathbf{x}) = \mathbf{W}_L\sigma_{L-1}(\mathbf{W}_{L-1}\sigma(\cdots\sigma(\mathbf{W}_1\mathbf{x})\cdots)$$

Width and Depth

- **Width** is the number of neurons in each layer (typically the widest)
- **Depth** is the number of layers
  - e.g., a 3-layer neural network with width 4
    (alternatively, a width 4 network with two hidden layers)

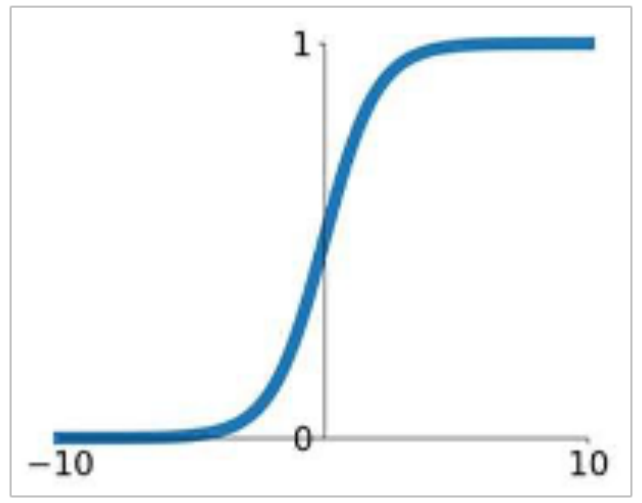


Activation functions

- There are many, for good reasons...
  - Two big categories: Saturating & Non-saturating

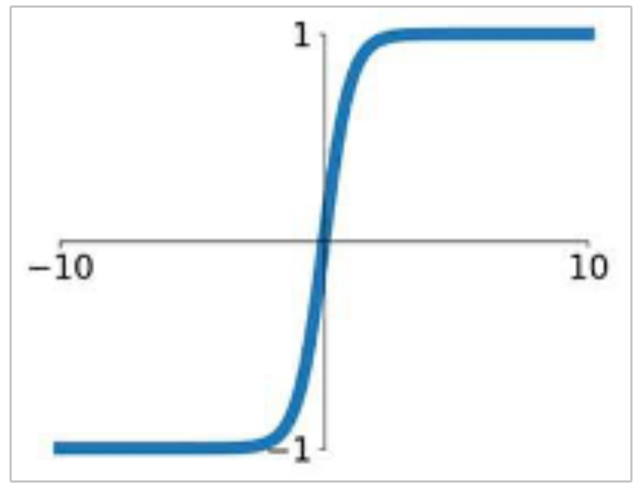
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



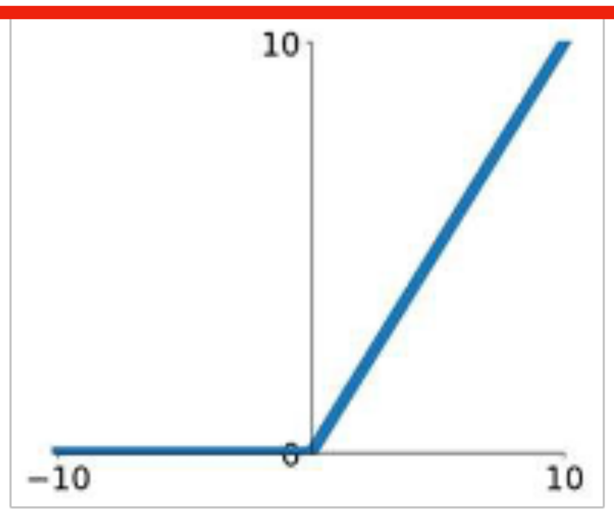
Tanh

$\tanh(x)$



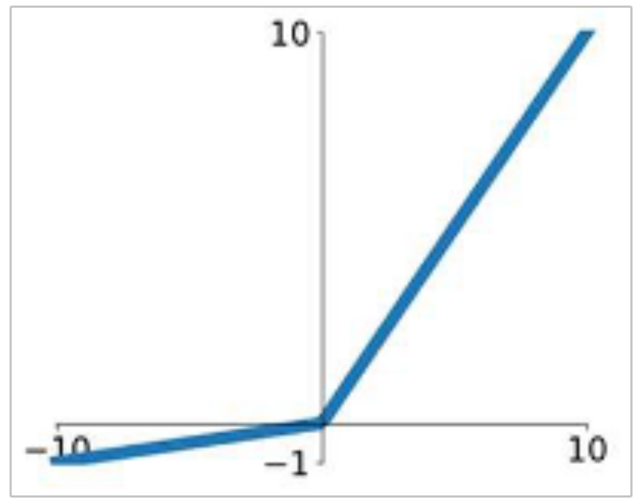
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

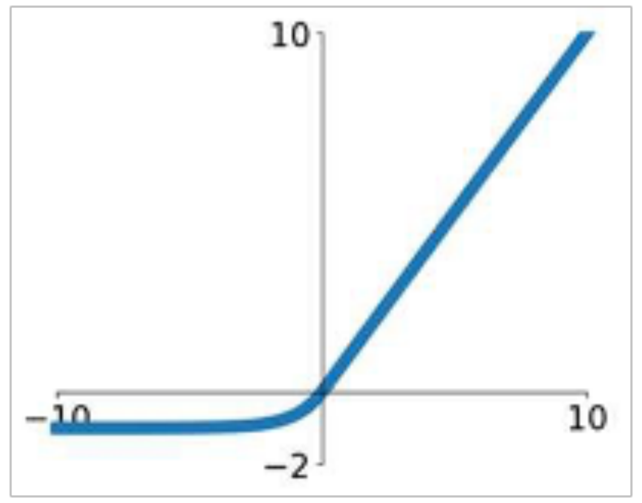


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



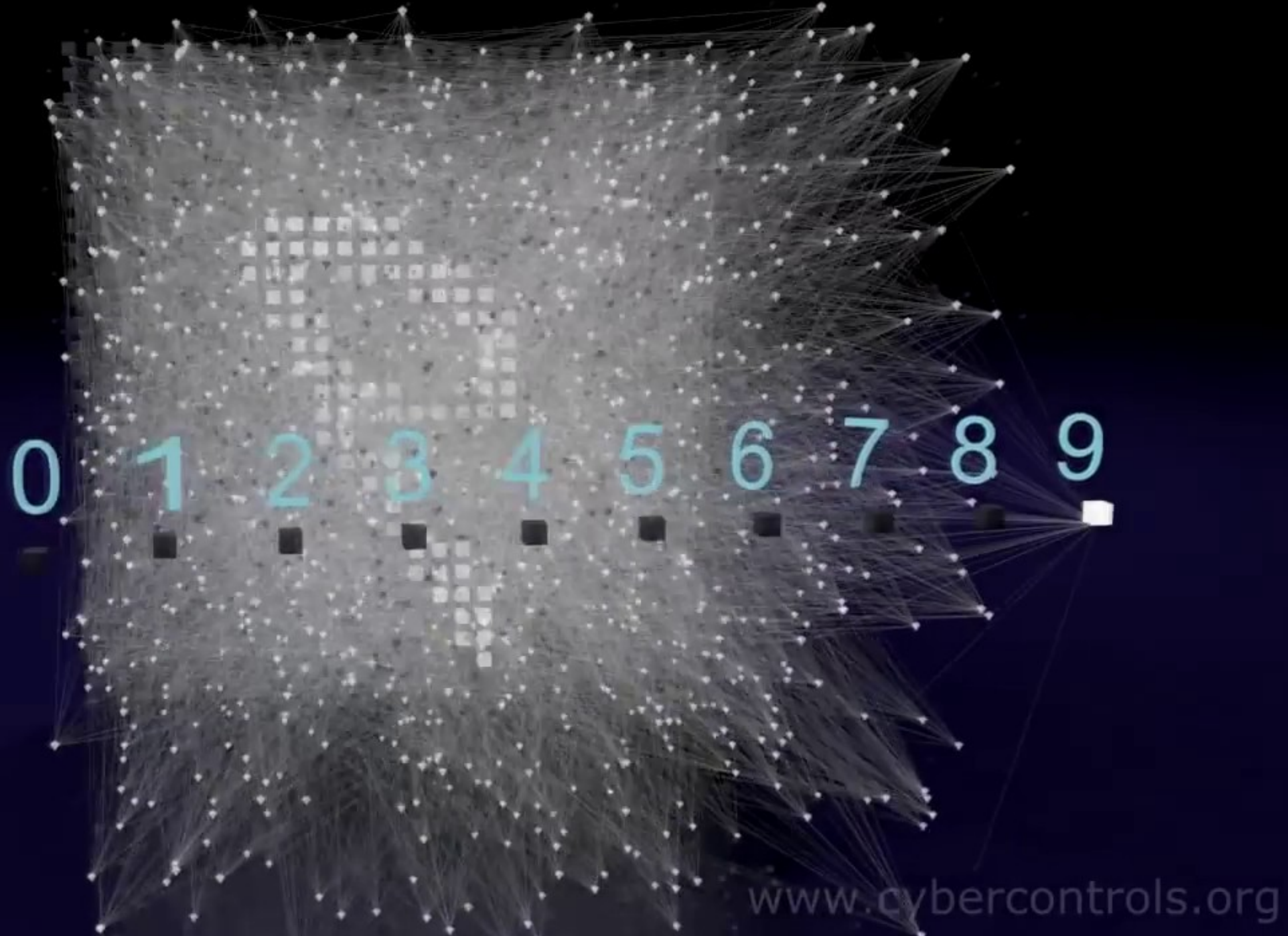
Activation functions

- **Q.** What happens without activation functions?

$$f(\mathbf{x}) = \mathbf{W}_L \mathbf{W}_{L-1} \cdots \mathbf{W}_1 \mathbf{x}$$
$$= \tilde{\mathbf{W}} \mathbf{x}$$

  - Equivalent to a linear function!
    (thus no merit)

Type: Perceptron
Data Set: MNIST
Hidden Neurons: 2000
Synapses: 1191000
Synapses shown: 2%
Learning: WCor



Why are deep neural networks cool?

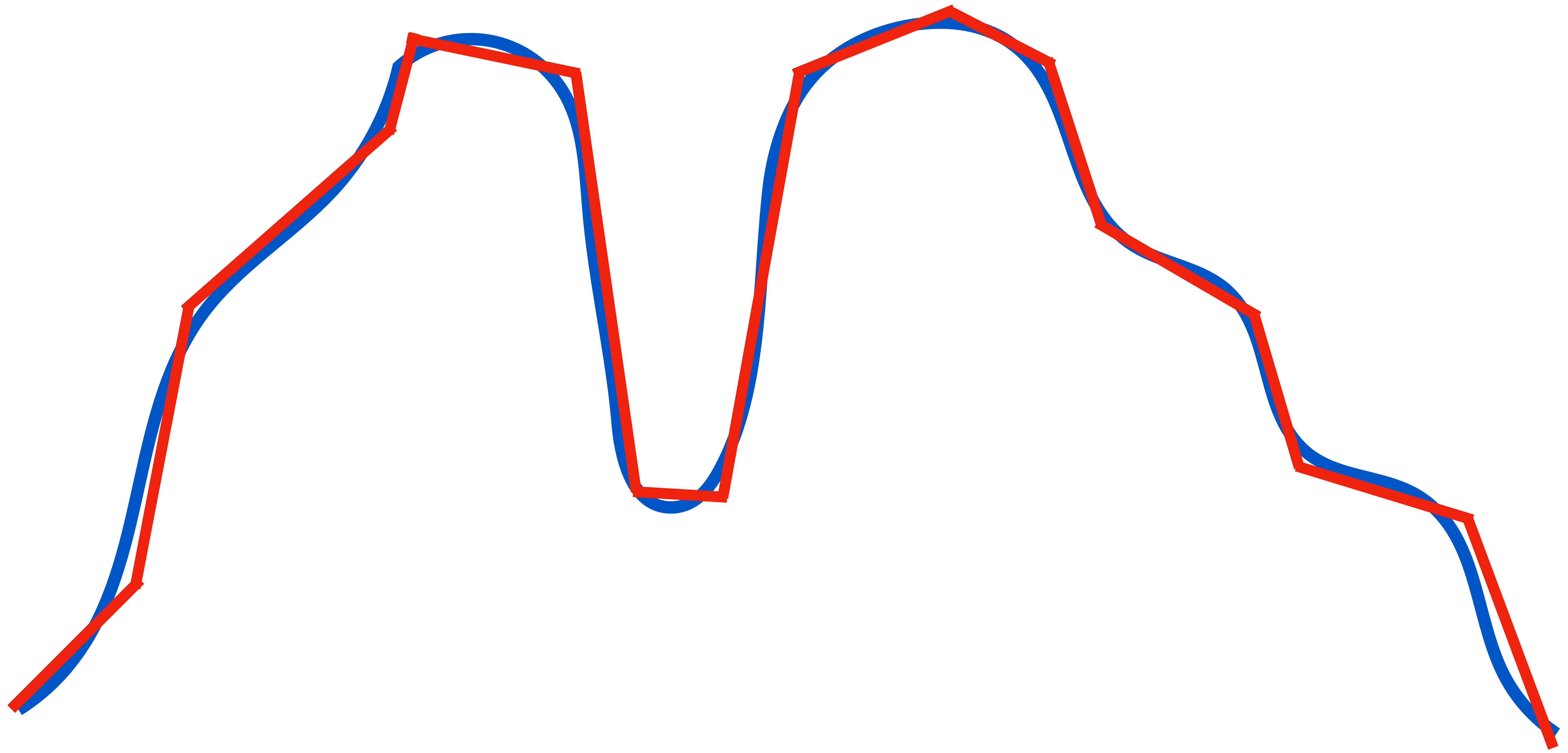
- Theoretically, this can represent **any continuous function**!
  (via so-called "universal approximation theorems")
  - Only requires one hidden layer, given sufficient width
- Easy to compute; mostly linear operations
  - Admits **parallel computation**
- Very **flexible** in size, and very **modular**.
  - Design new operations and combine

Universal Approximation Theorem (rough)

Theorem. Given any function $g(\cdot)$ and $\epsilon > 0$,

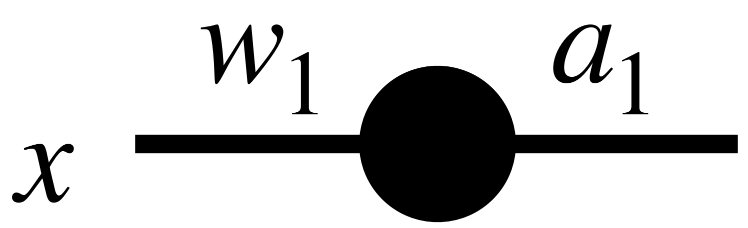
one can find a two-layer **ReLU** neural network $f(\cdot)$ such that

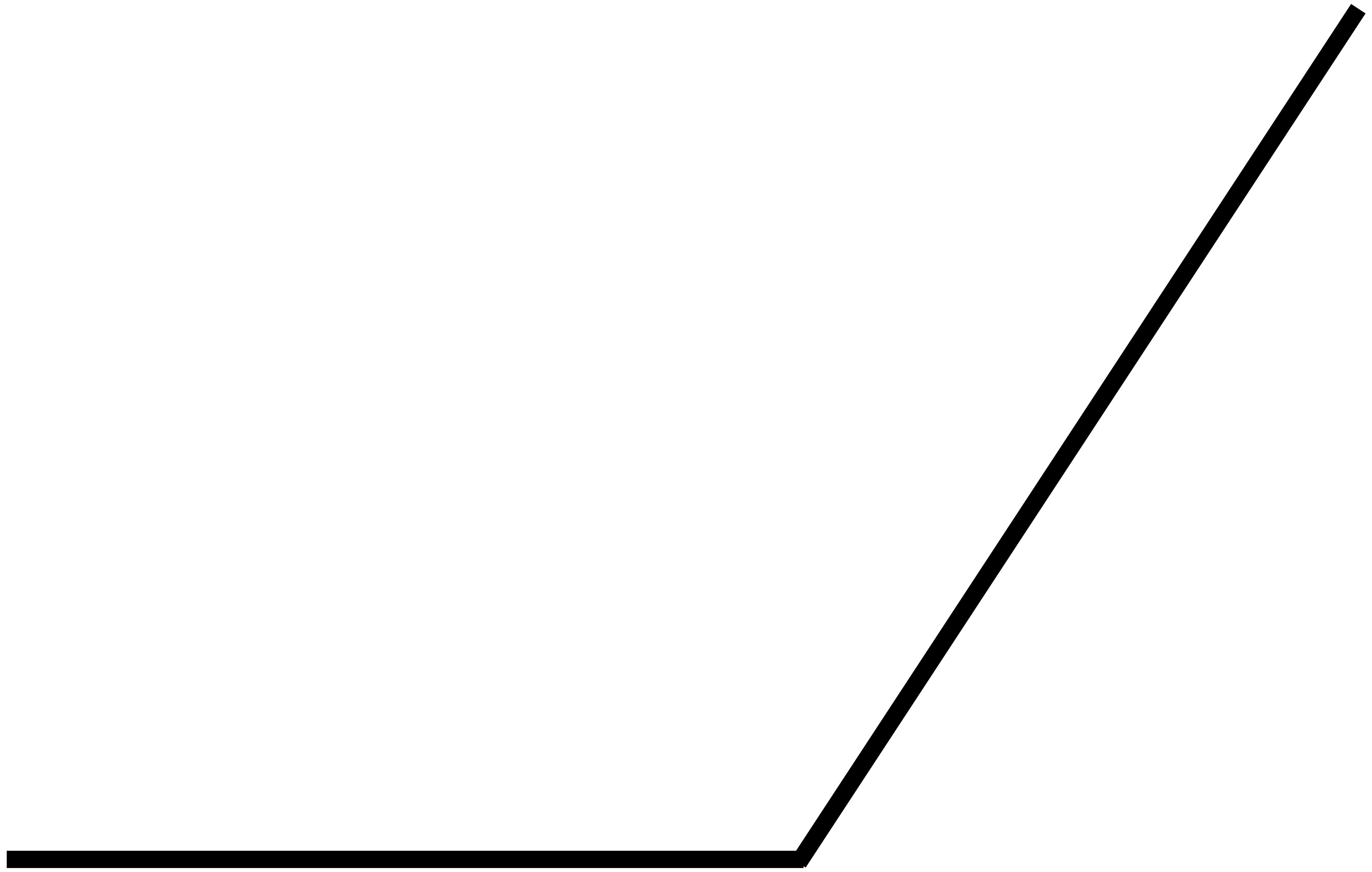
$$\sup_{x \in [0,1]} |g(x) - f(x)| \leq \epsilon.$$



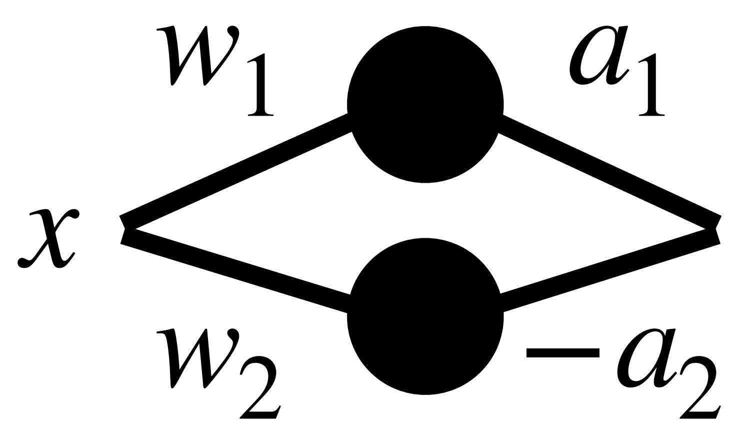
Proof idea

A single ReLU neuron looks like this.

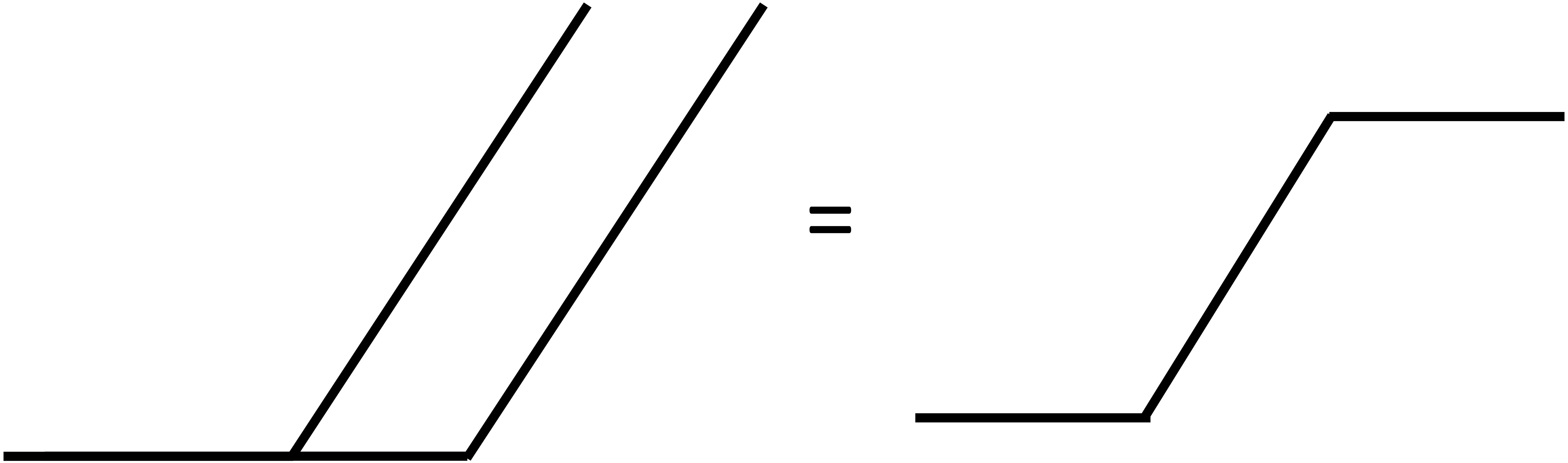




Proof idea

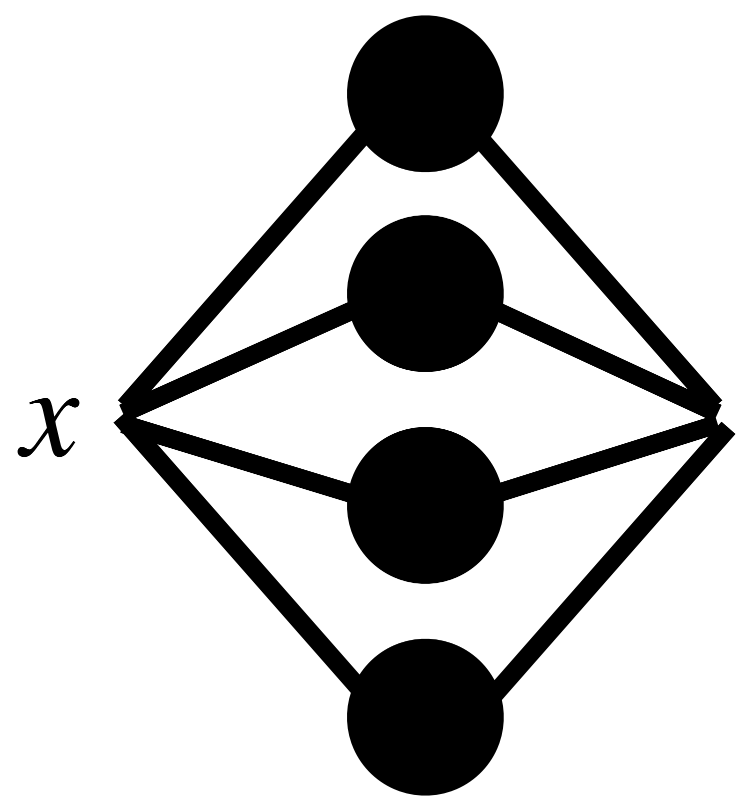


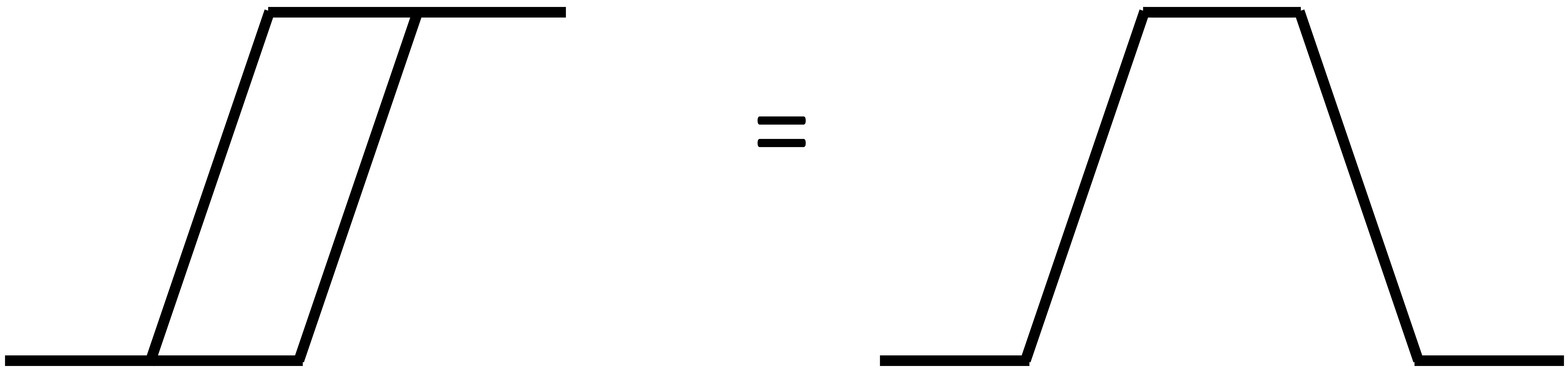
Difference of two single neurons makes the "hard sigmoid"



Proof idea

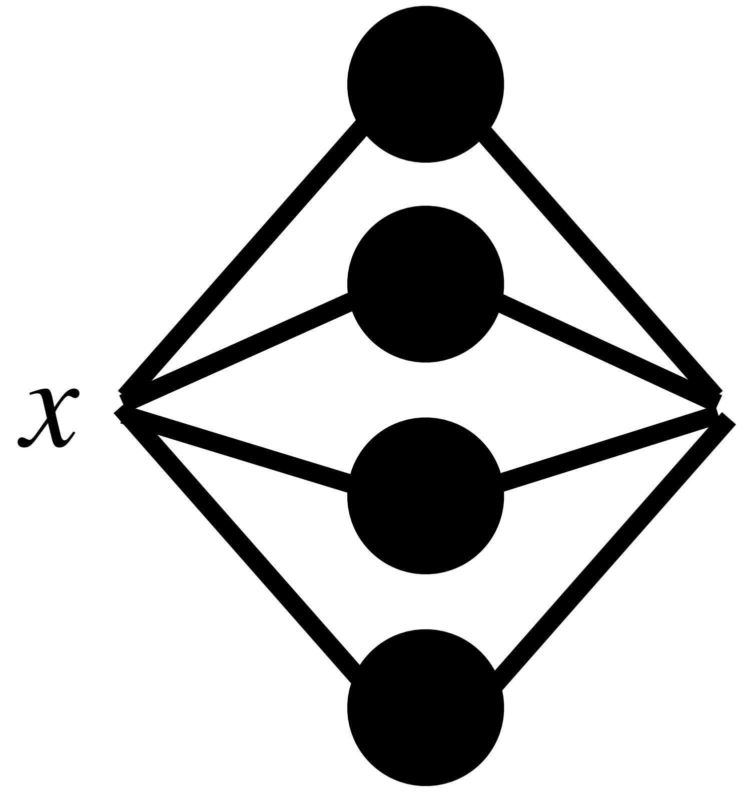
Difference of two hard sigmoids makes a "bump"

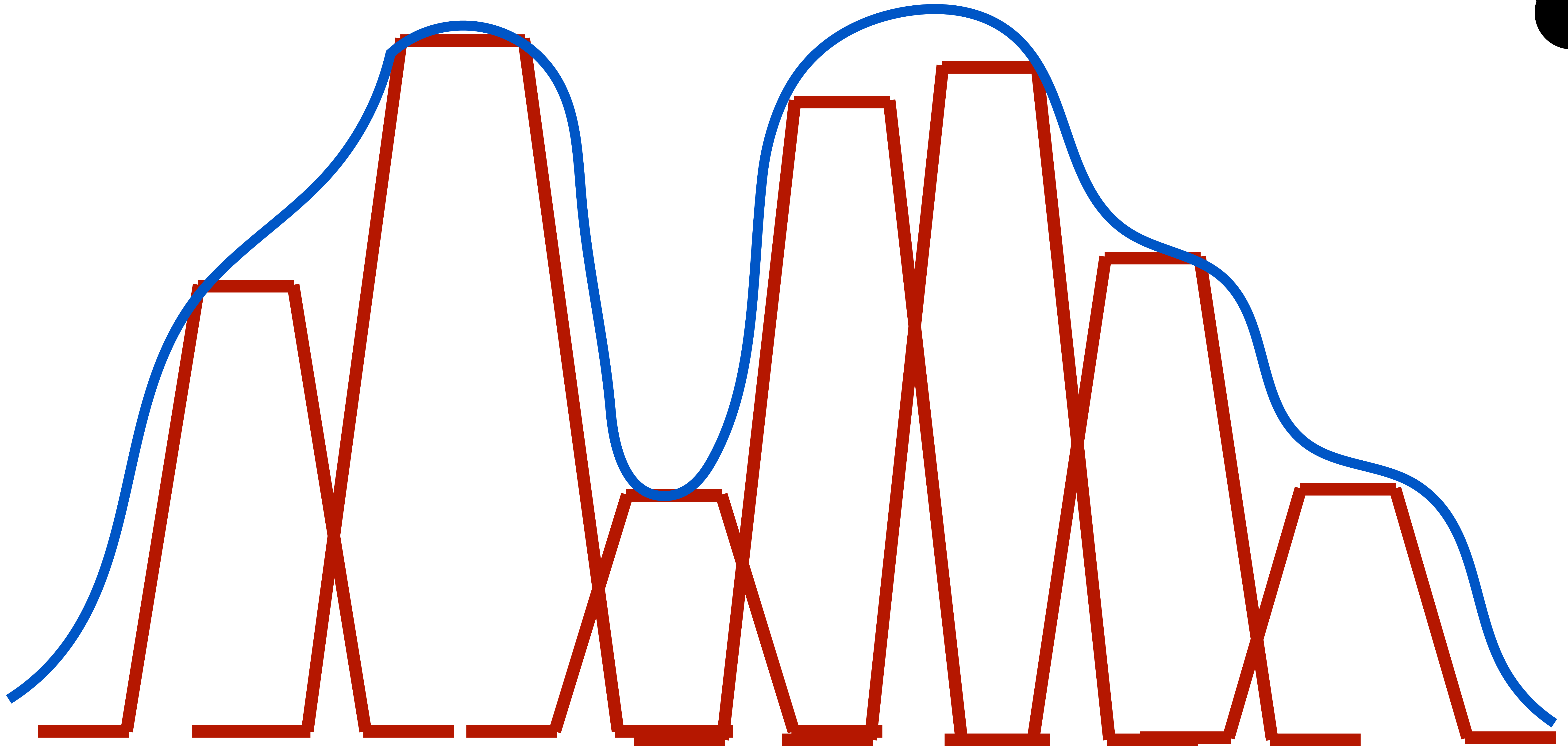




Proof idea

Use bumps to approximate the target function





Cheers

- *Next up.* Convolutional layers