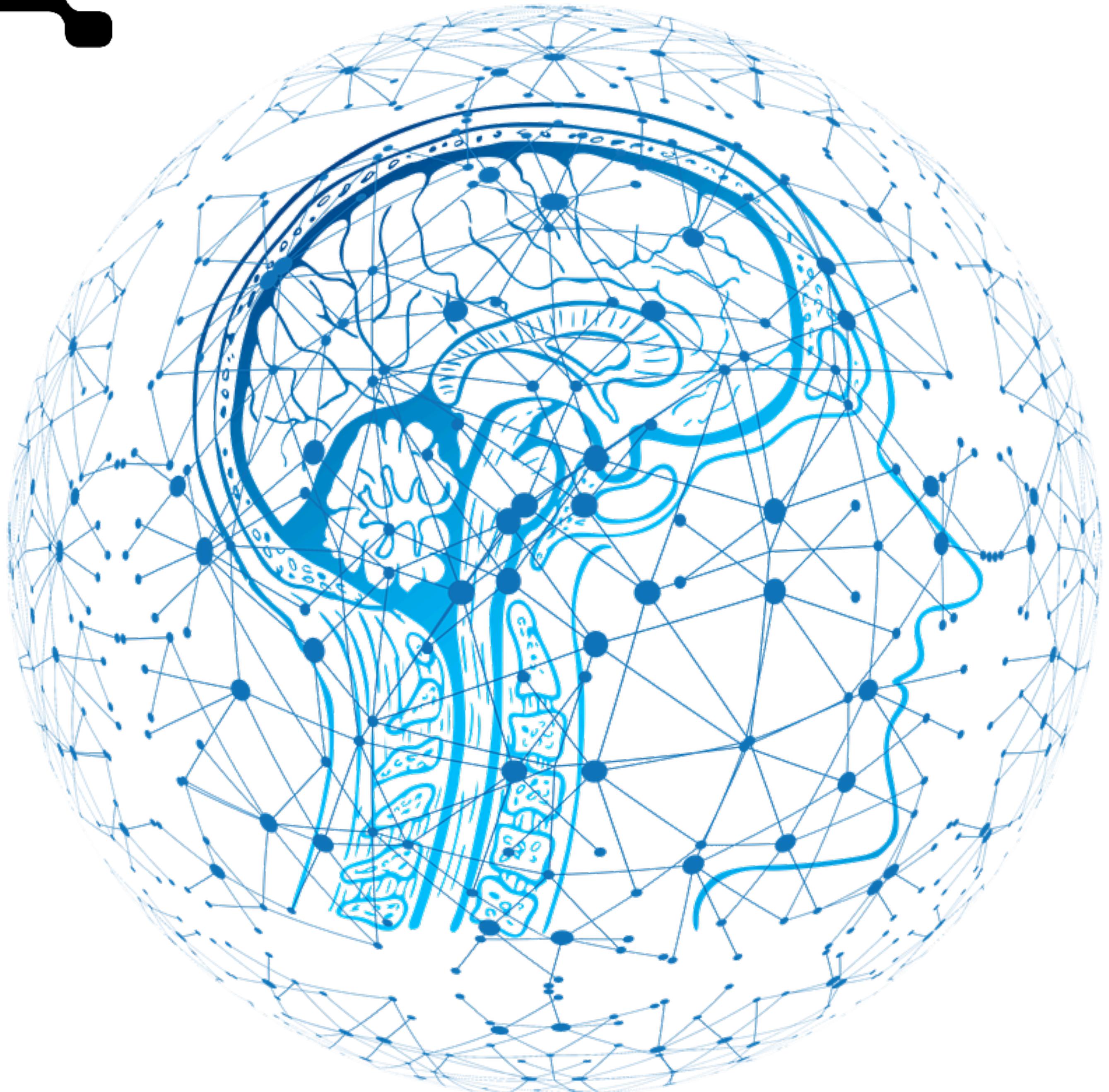


# Introduction to Machine Learning & Deep Learning

- Main concepts in artificial intelligence and machine learning
- Types of learning and available libraries
- Hands-on exercises on CNNs and RNNs

## Programme

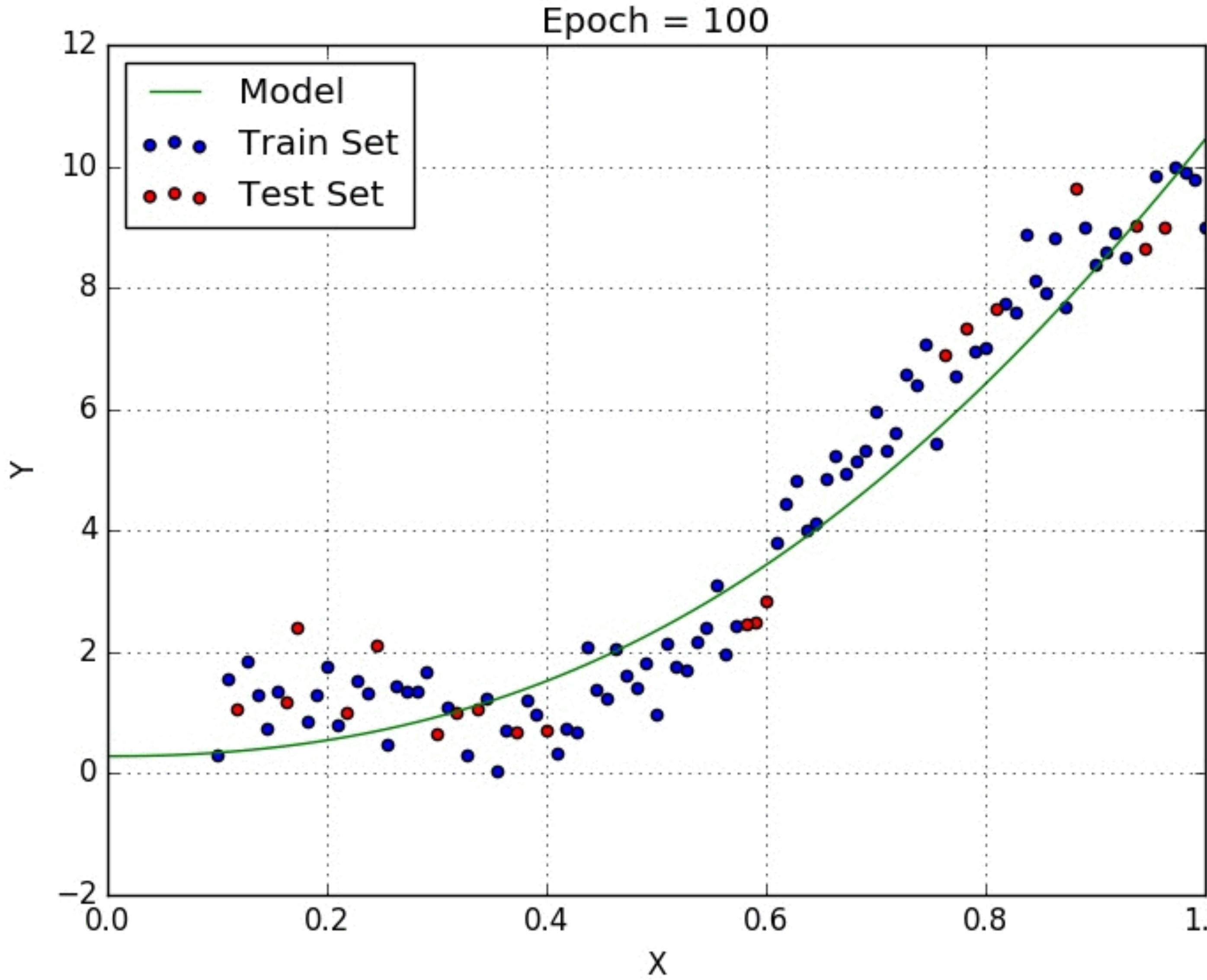
09:00-11:00: Introduction ML/DL  
11:00-11:30: break  
11:30-12:30: Data Processing  
12:30-13:30: Lunch break  
13:30-14:30: Hands-on session  
14:30-14:45: break  
14:45-16:00: Hands-on session



# Introduction to Machine Learning & Deep Learning

*Maxwell Cai (SURFsara)*

# What *is* machine learning?

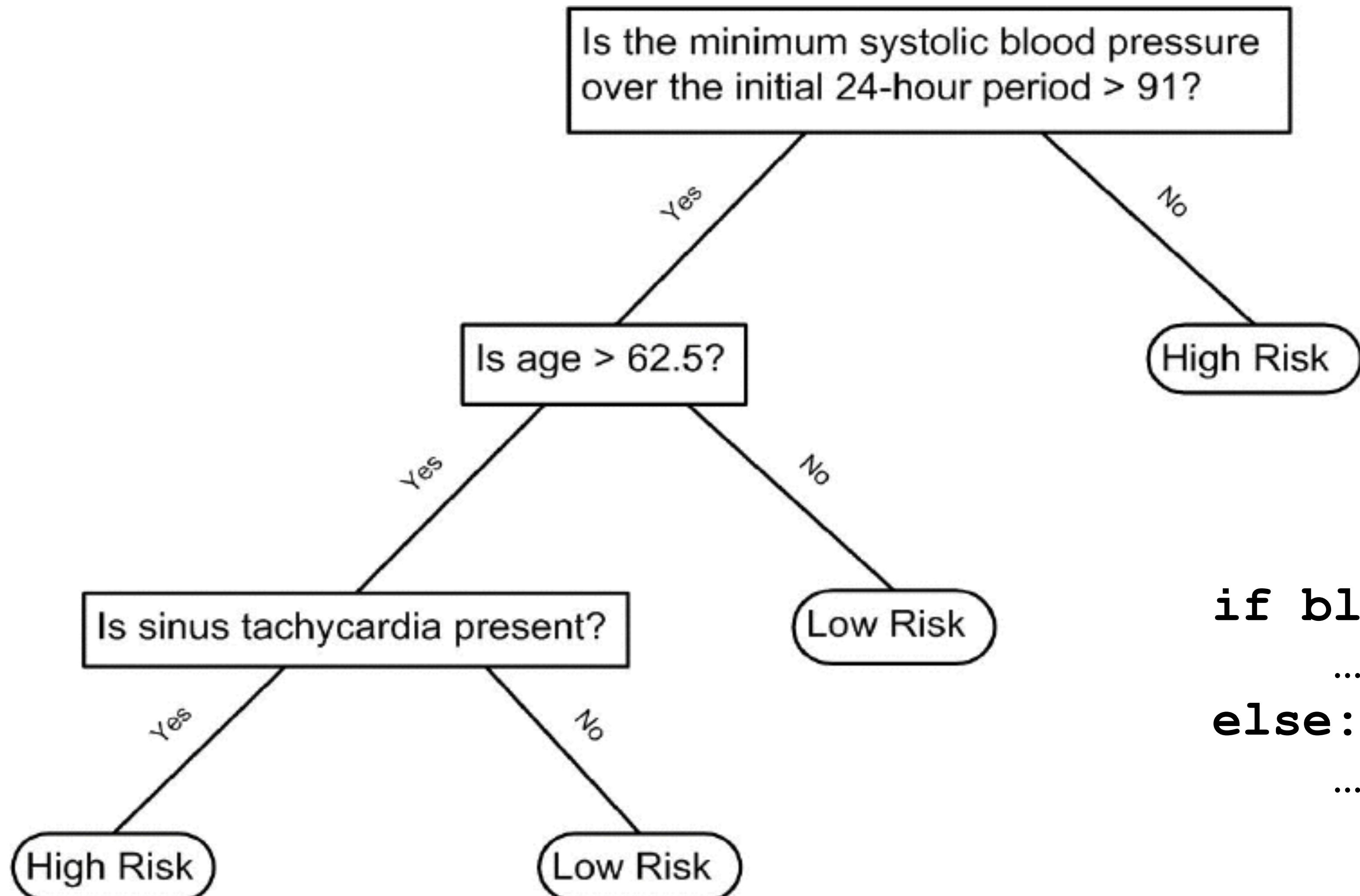


**Could be as simple as curve fitting!**

**Typical applications in general:**

- Classification
- Regression
- Dimensionality reduction
- Control

# Decision Tree(s)



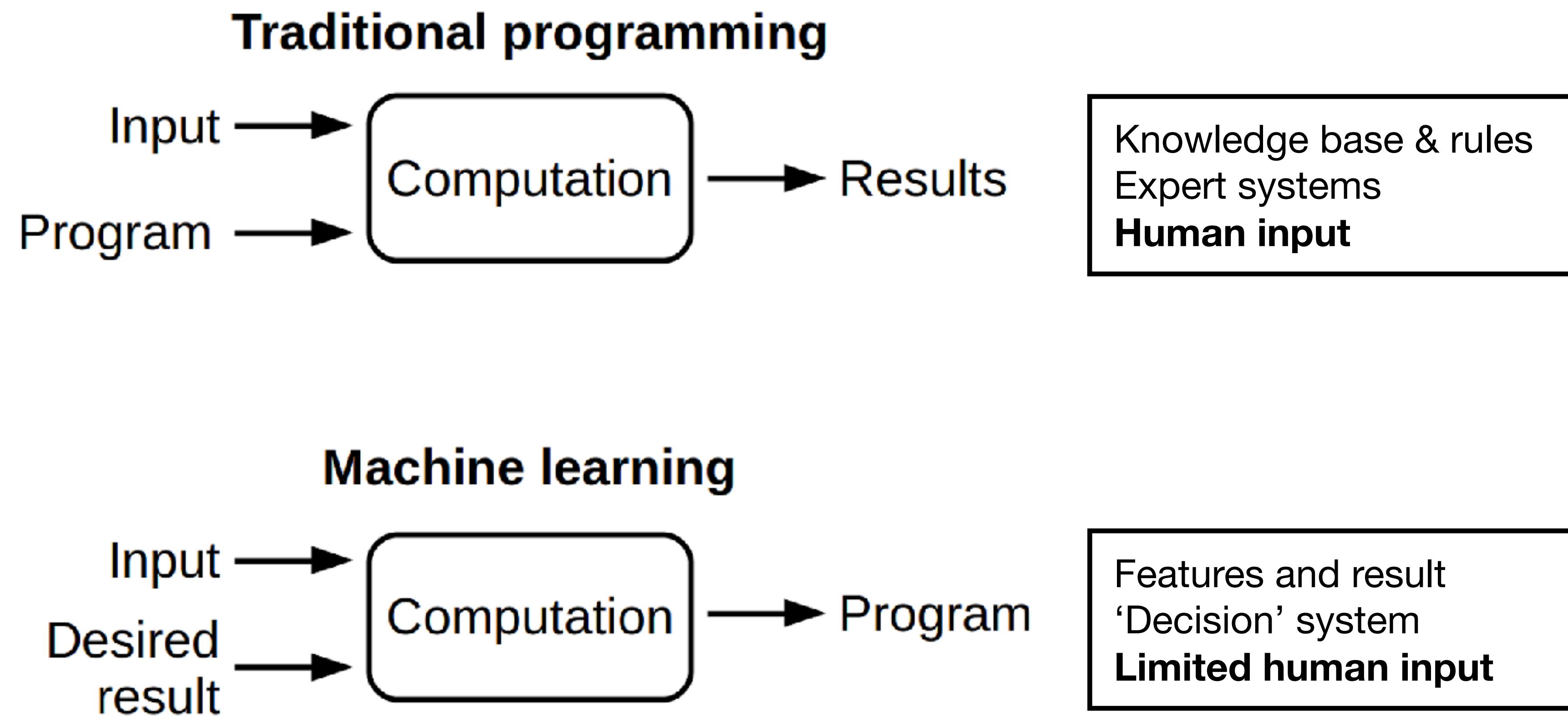
```
if blood_pressure > 91:
```

...

```
else:
```

...

# What *is* machine learning?



# AI vs ML vs DL

## ARTIFICIAL INTELLIGENCE

Any technique that enables computers to mimic human behavior



## MACHINE LEARNING

Ability to learn without explicitly being programmed

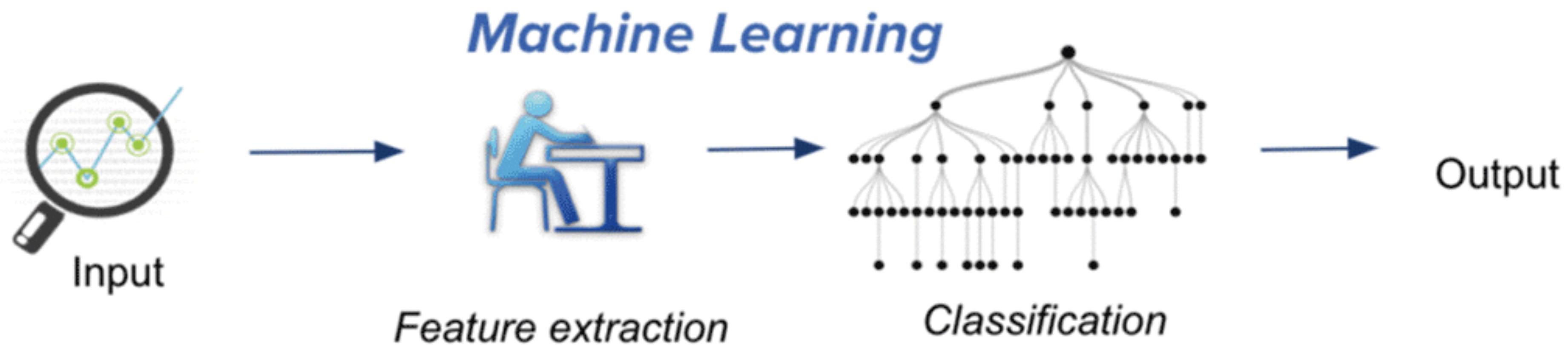


## DEEP LEARNING

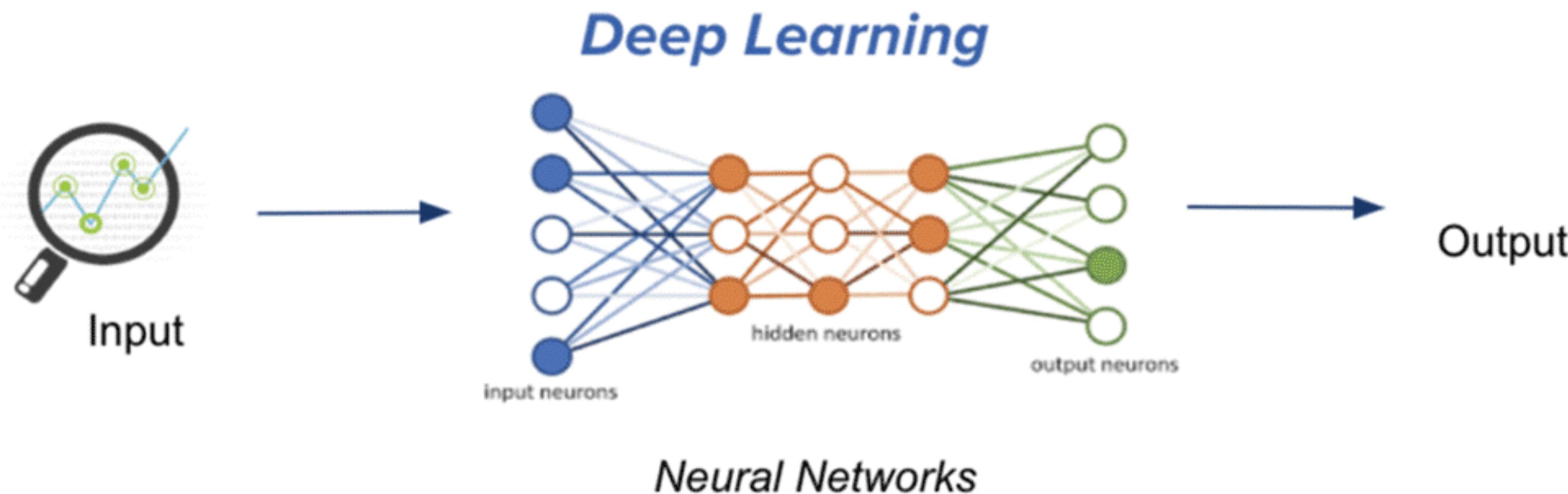
Extract patterns from data using neural networks



Credit: Alexander Amini/MIT



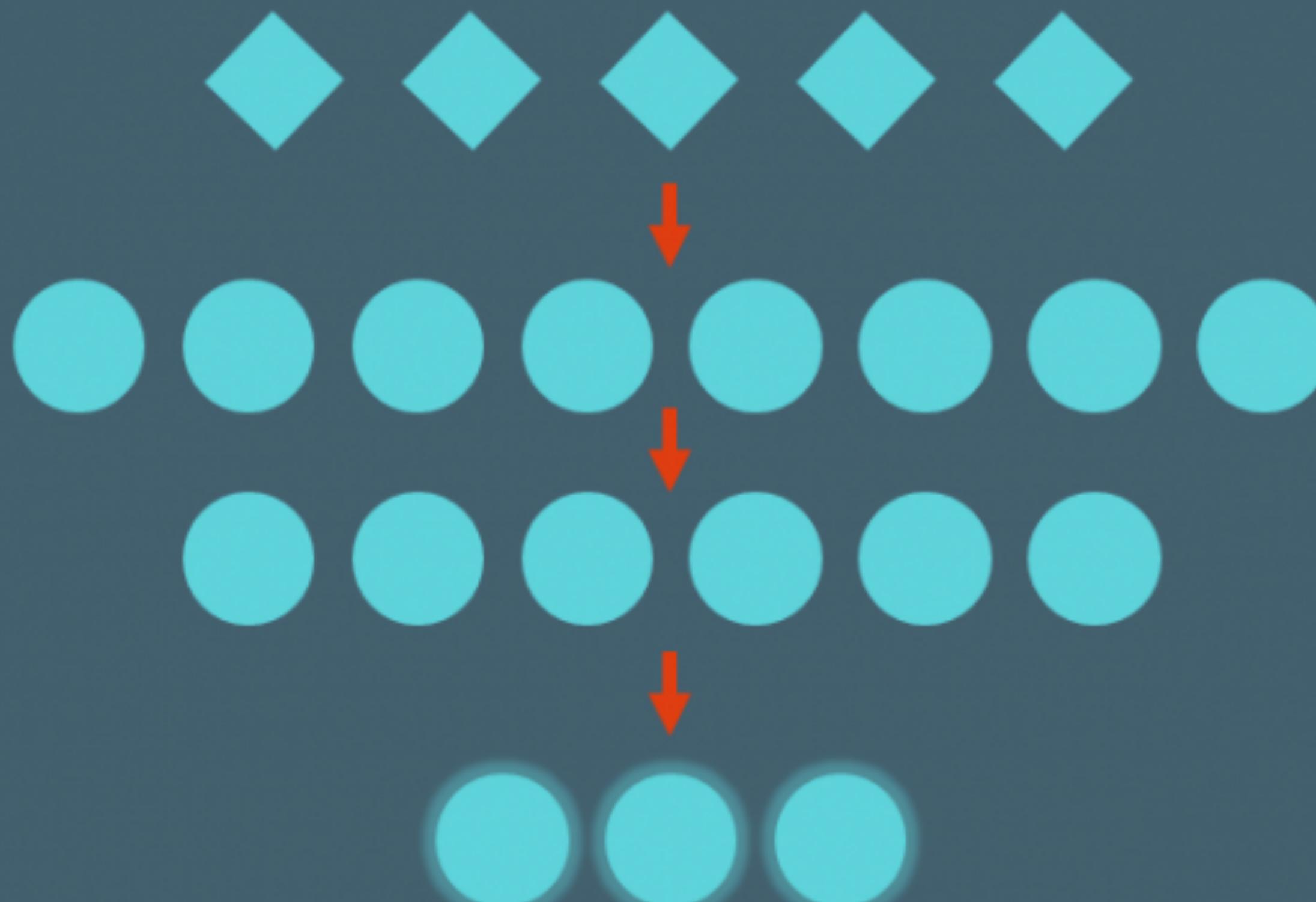
Traditional machine learning uses hand-crafted features, which is tedious and costly to develop.



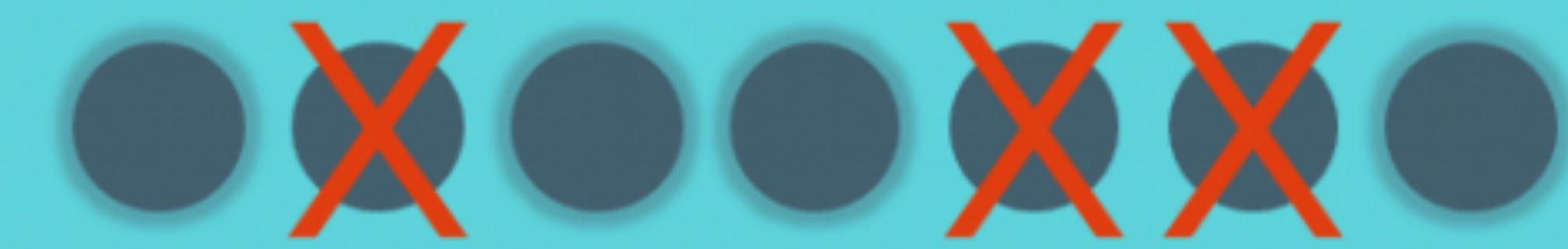
Deep learning learns hierarchical representation from the data itself, and scales with more data.

# Feature Extraction & Selection

feature extraction



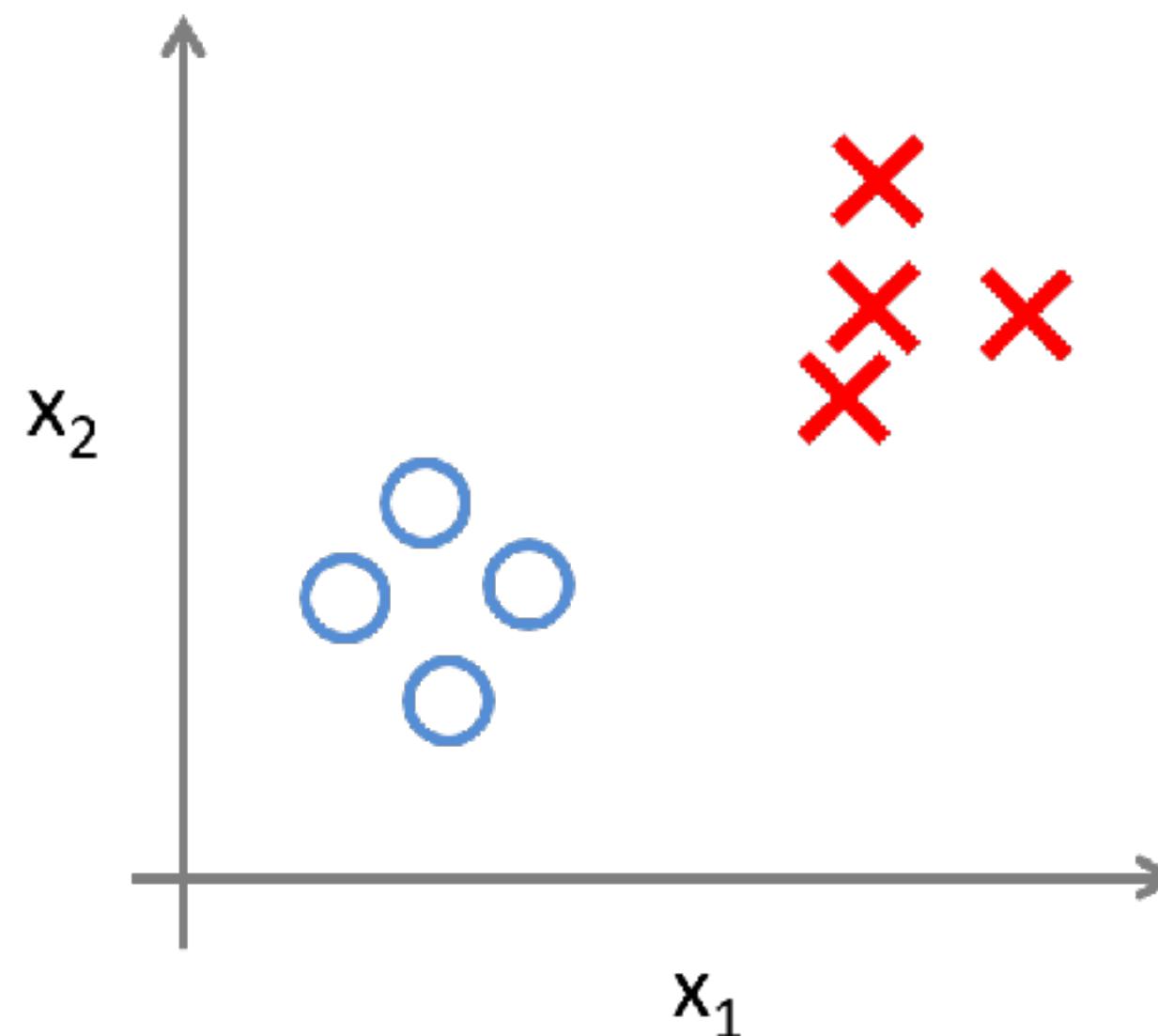
feature selection



# Categories of machine learning

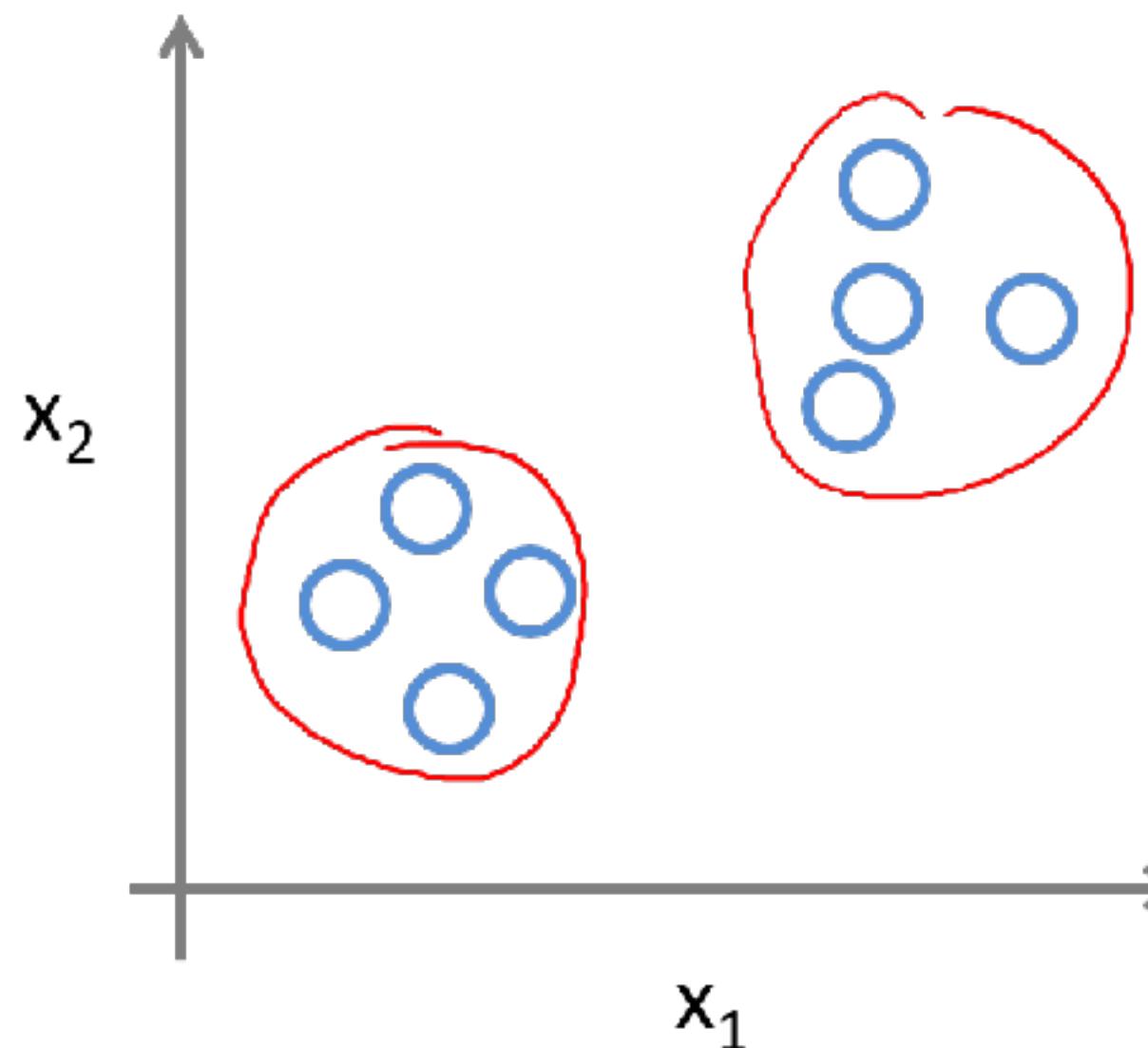
Supervised

Learn from the labels



Unsupervised

Detect patterns in the data

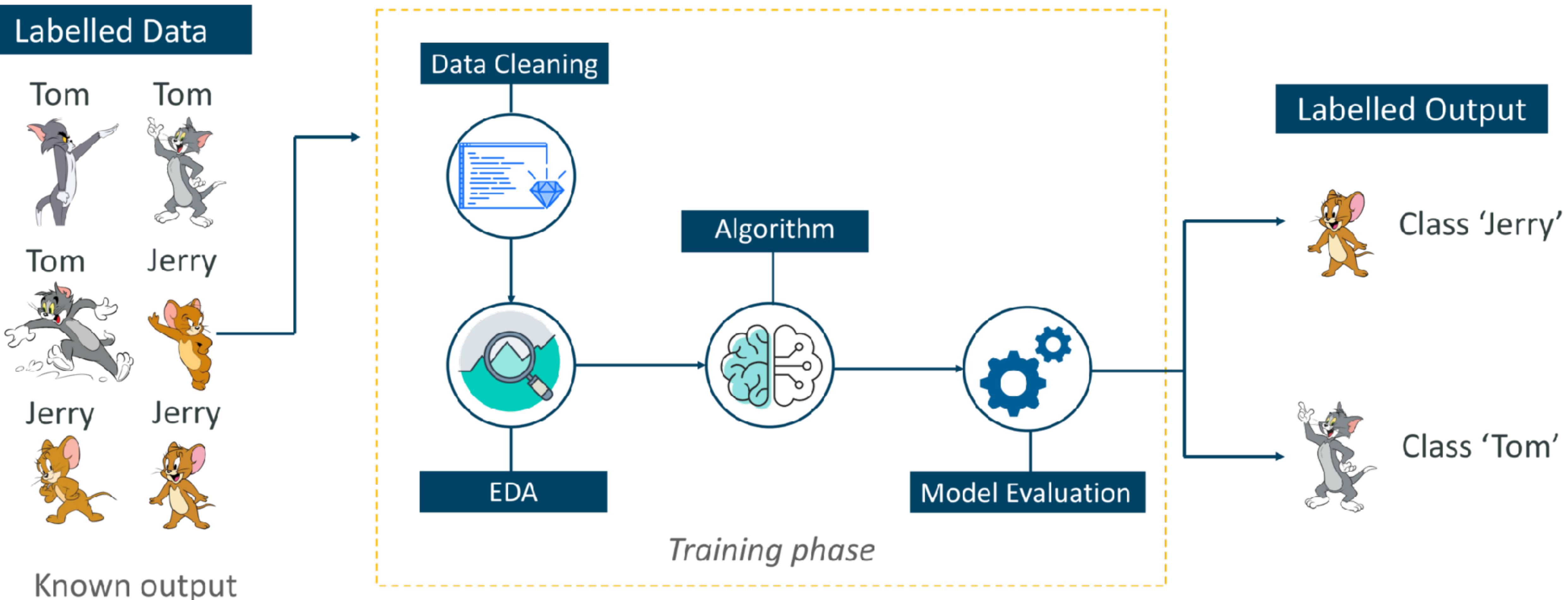


Reinforcement

Learn from mistakes



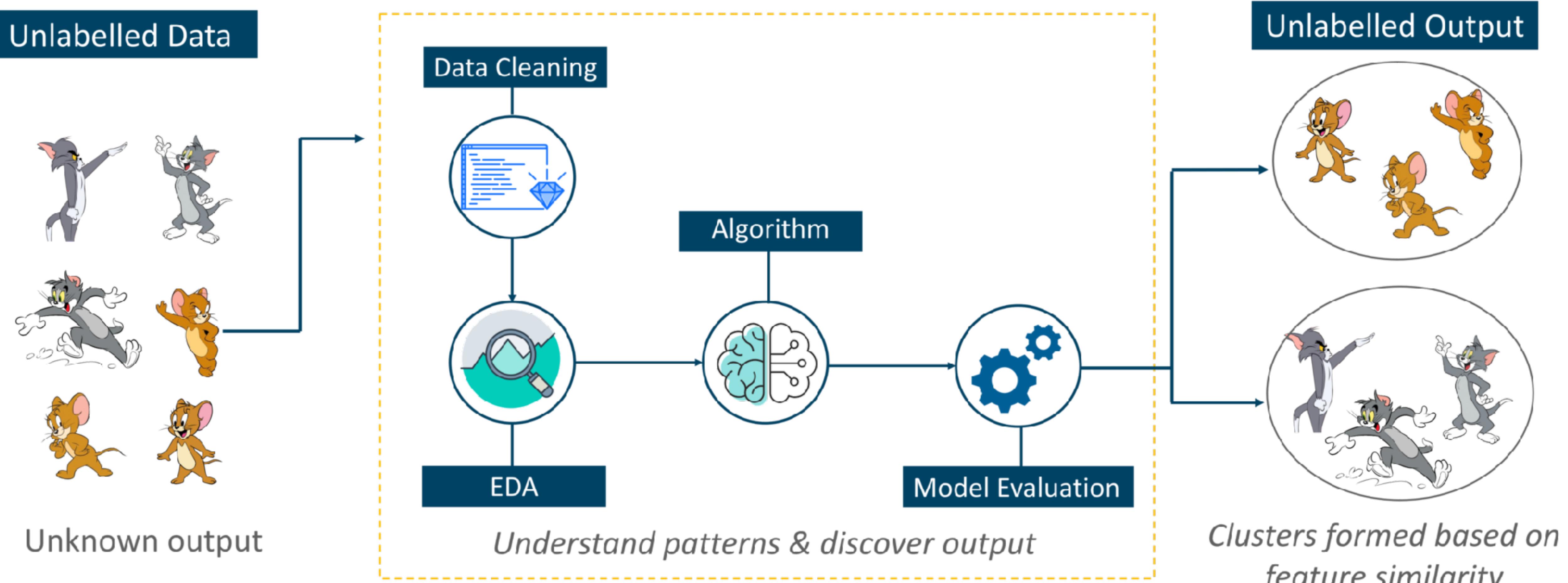
# Supervised Learning



EDA: Exploratory Data Analysis

Source: edureka.co

# Unsupervised Learning



EDA: Exploratory Data Analysis

Source: edureka.co

# Reinforcement Learning



**Performance  
after 0  
episodes**

## Observation

Type: Box(4)

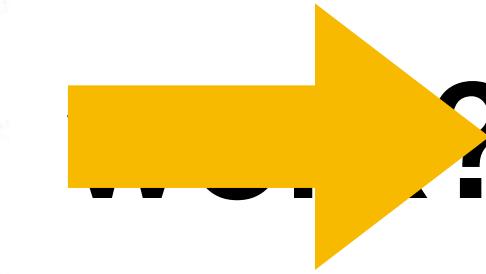
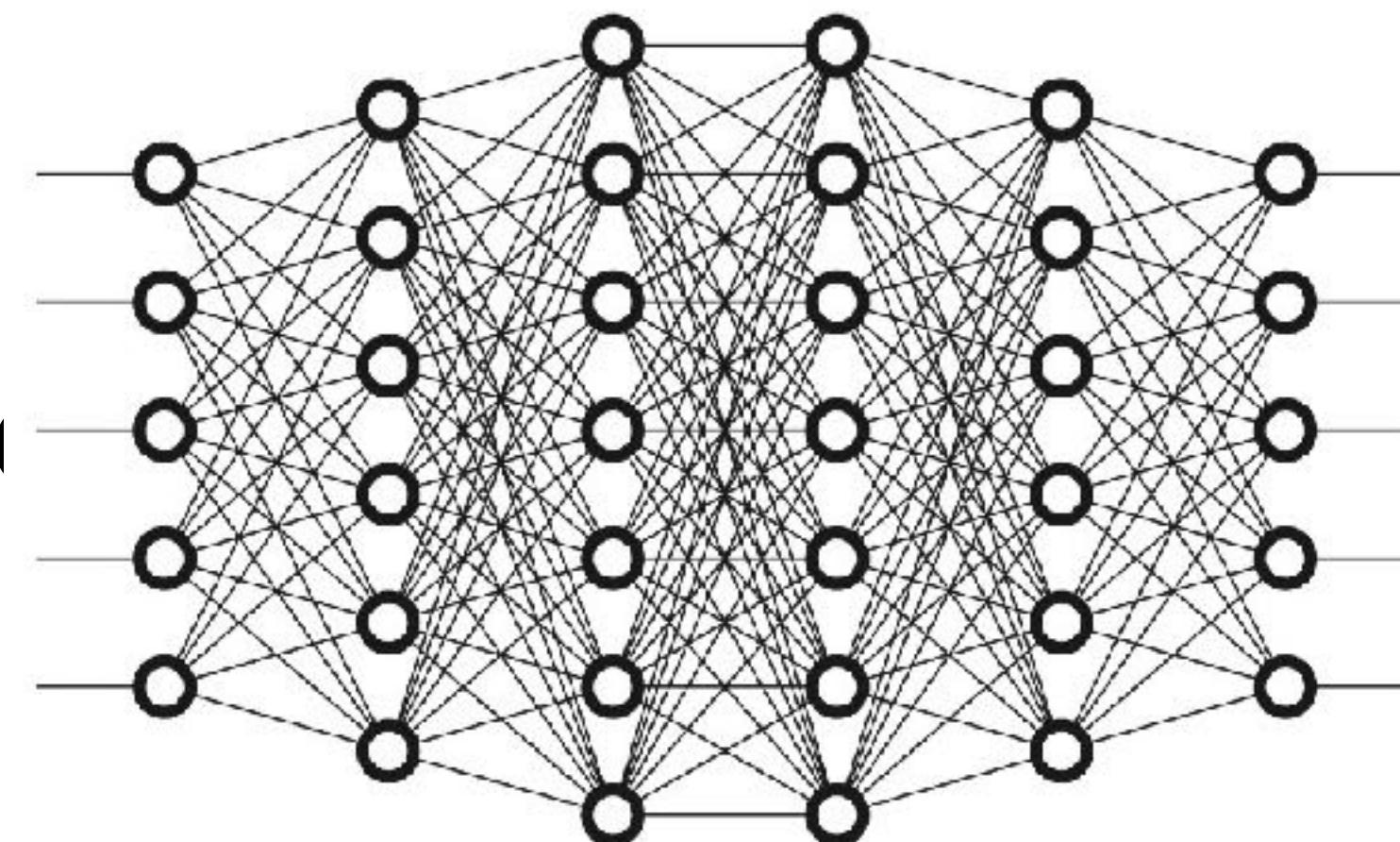
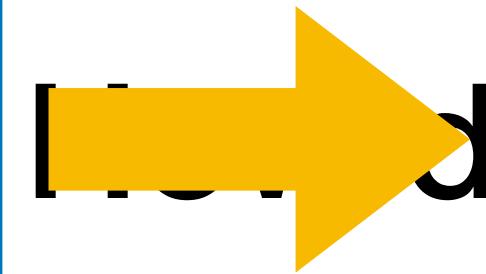
Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

## Actions

Type: Discrete(2)

Num	Action
0	Push cart to the left
1	Push cart to the right

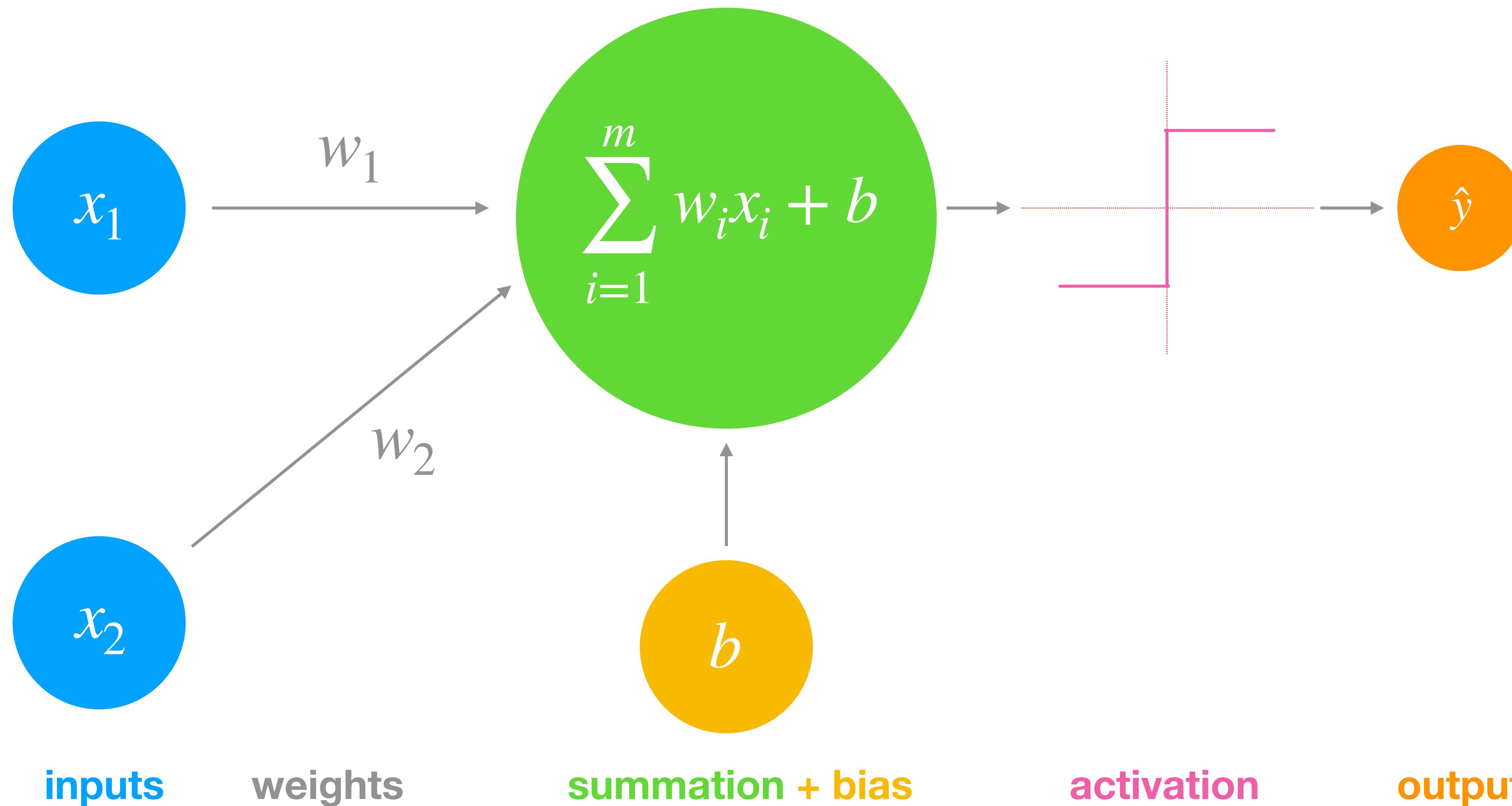
Input



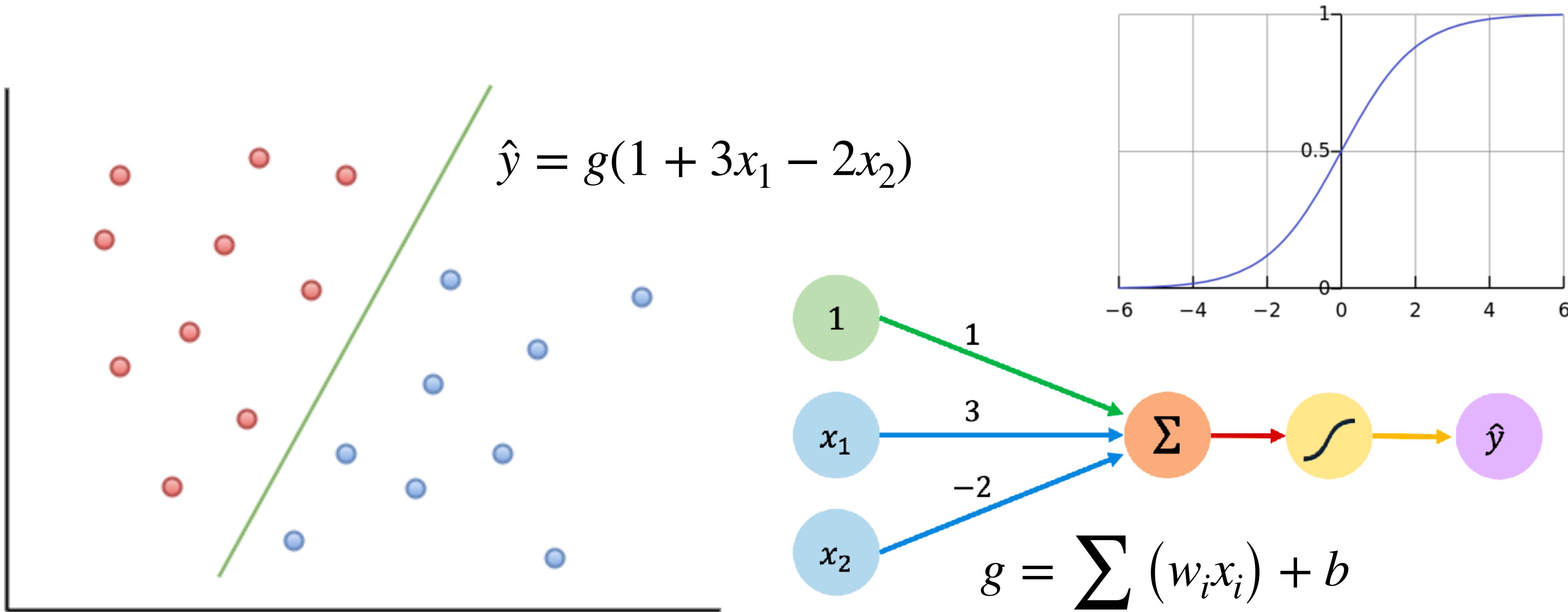
Output

$$\hat{y} = f_{\text{NN}}(x_1, x_2, \dots, x_n)$$

# Prediction



# Neurons/Perceptrons & Activation Functions



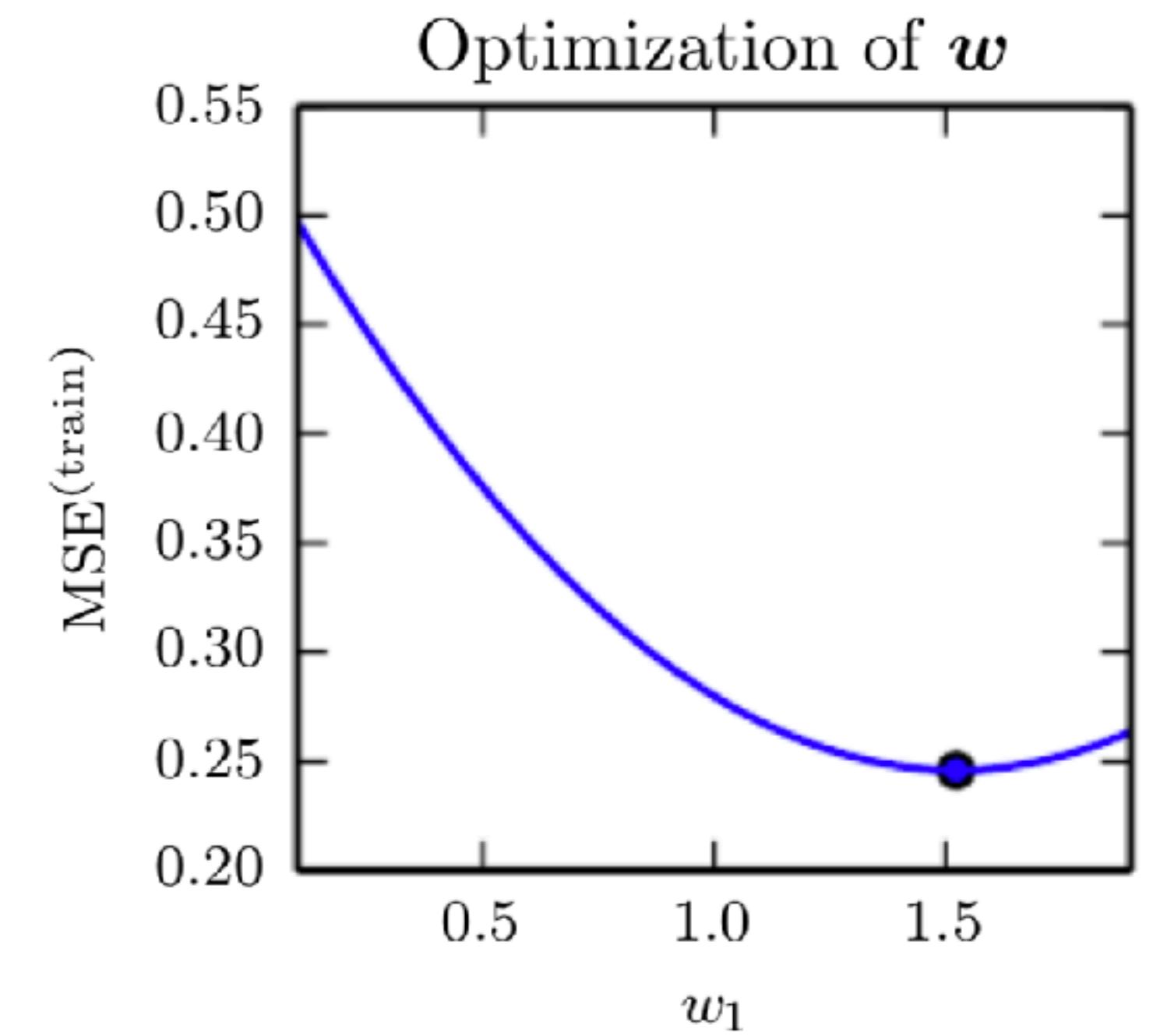
# A few more terms...

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b$$

Weights      Bias  
↓            ↓  
Estimate      Input features

Evaluation with the mean squared error

$$\text{MSE}(\mathbf{W}, b) = \frac{1}{N} \sum_{i=1}^N \left( y_i^{(\text{test})} - \hat{y}_i^{(\text{test})} \right)^2$$



What is the machine actually *learning*?

# Learning: Optimizing the loss function

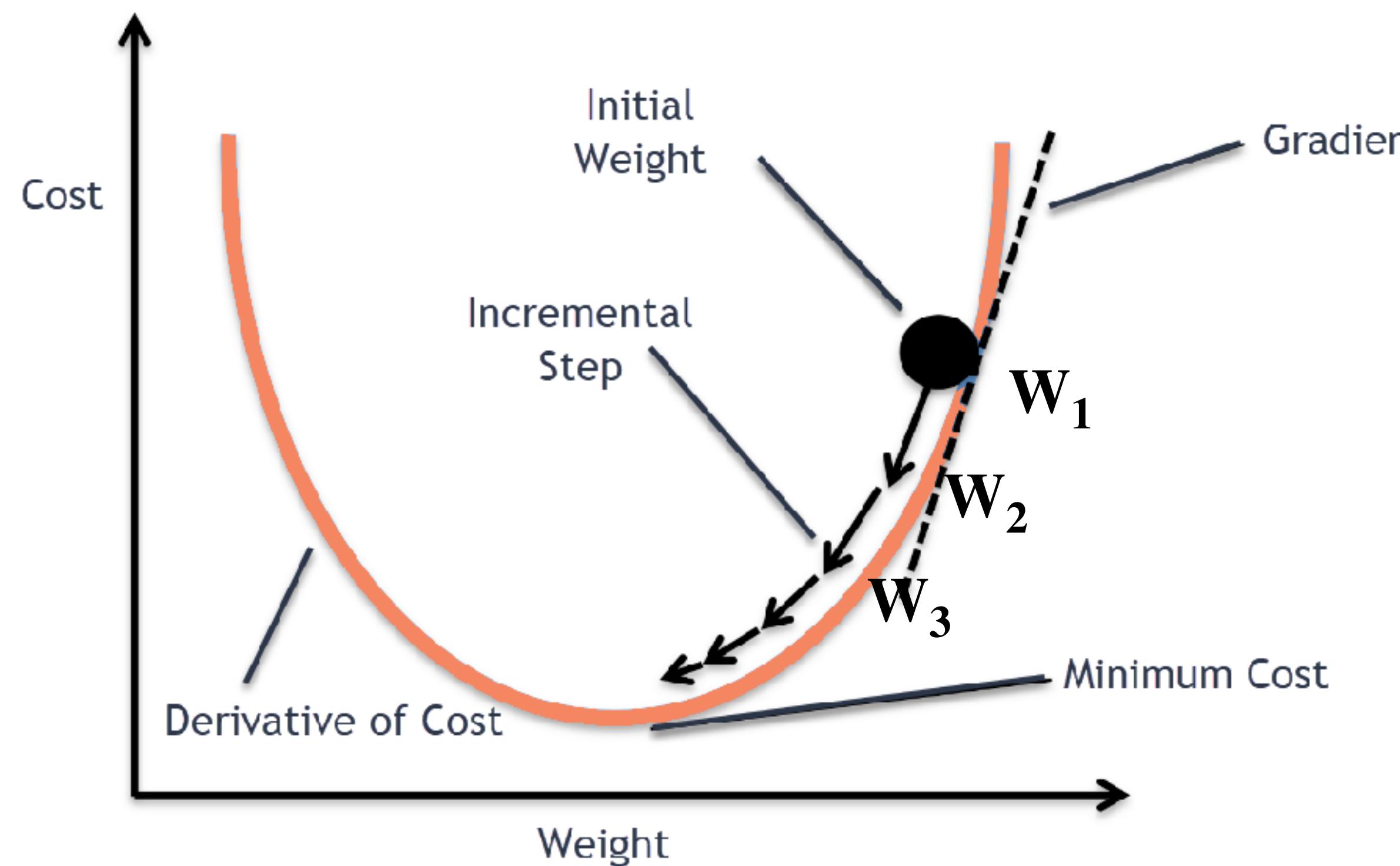
**Objective:** minimize the **difference** between predicted value the actual value

$$\mathcal{L}(\mathbf{W}, b) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

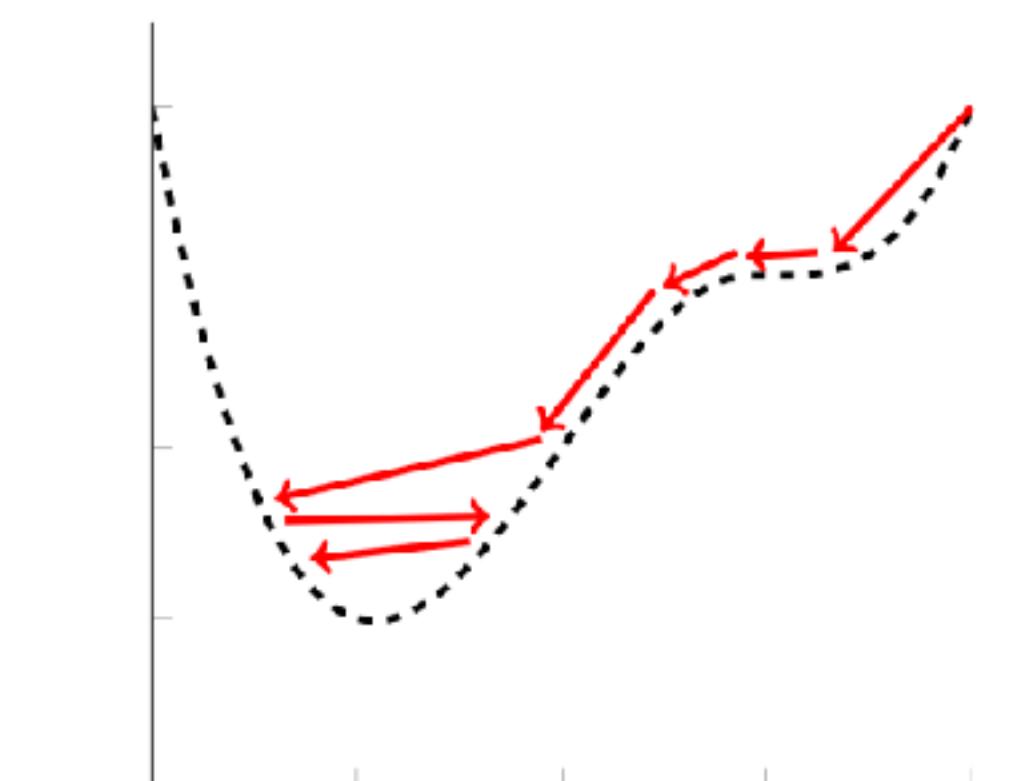
$\nabla \mathcal{L}(\mathbf{W}, b)$

$$\mathbf{W}_{j+1} = \mathbf{W}_j - \alpha \nabla \mathcal{L}(\mathbf{W}_j, b)$$

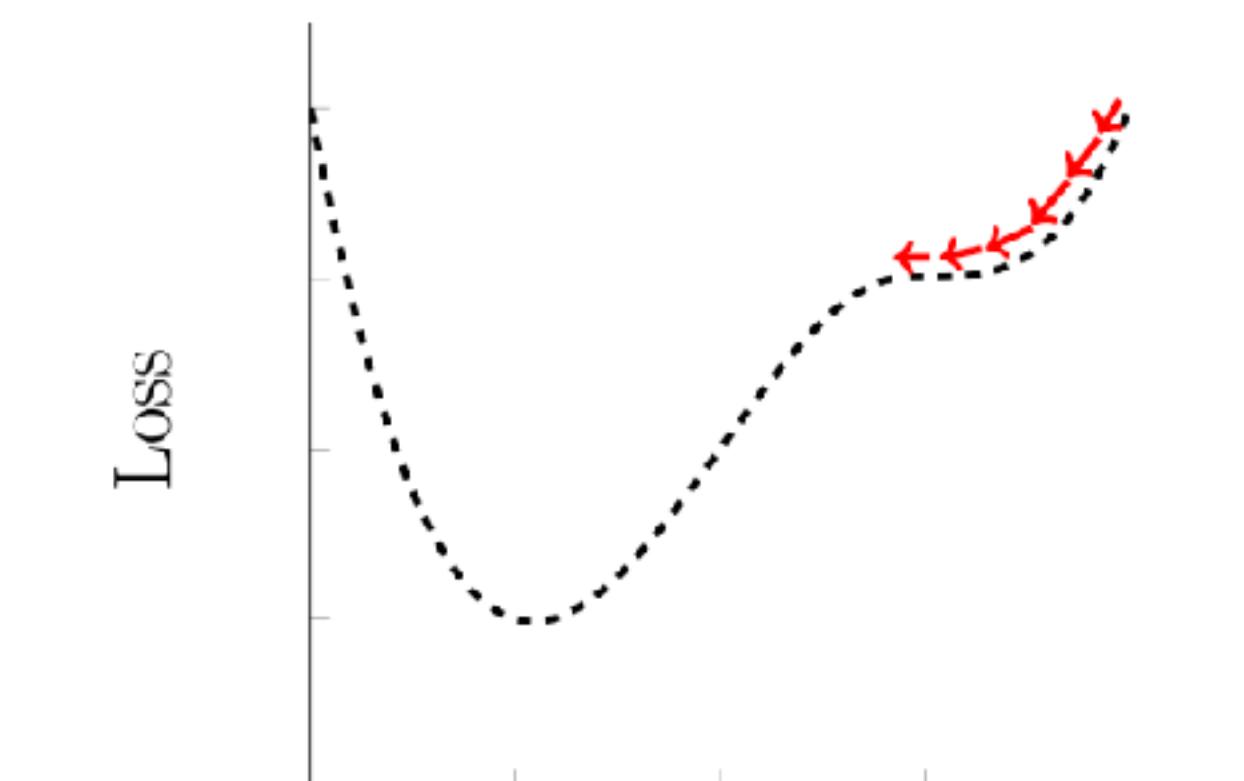
↑  
**Learning rate**



High Learning Rate



Low Learning Rate

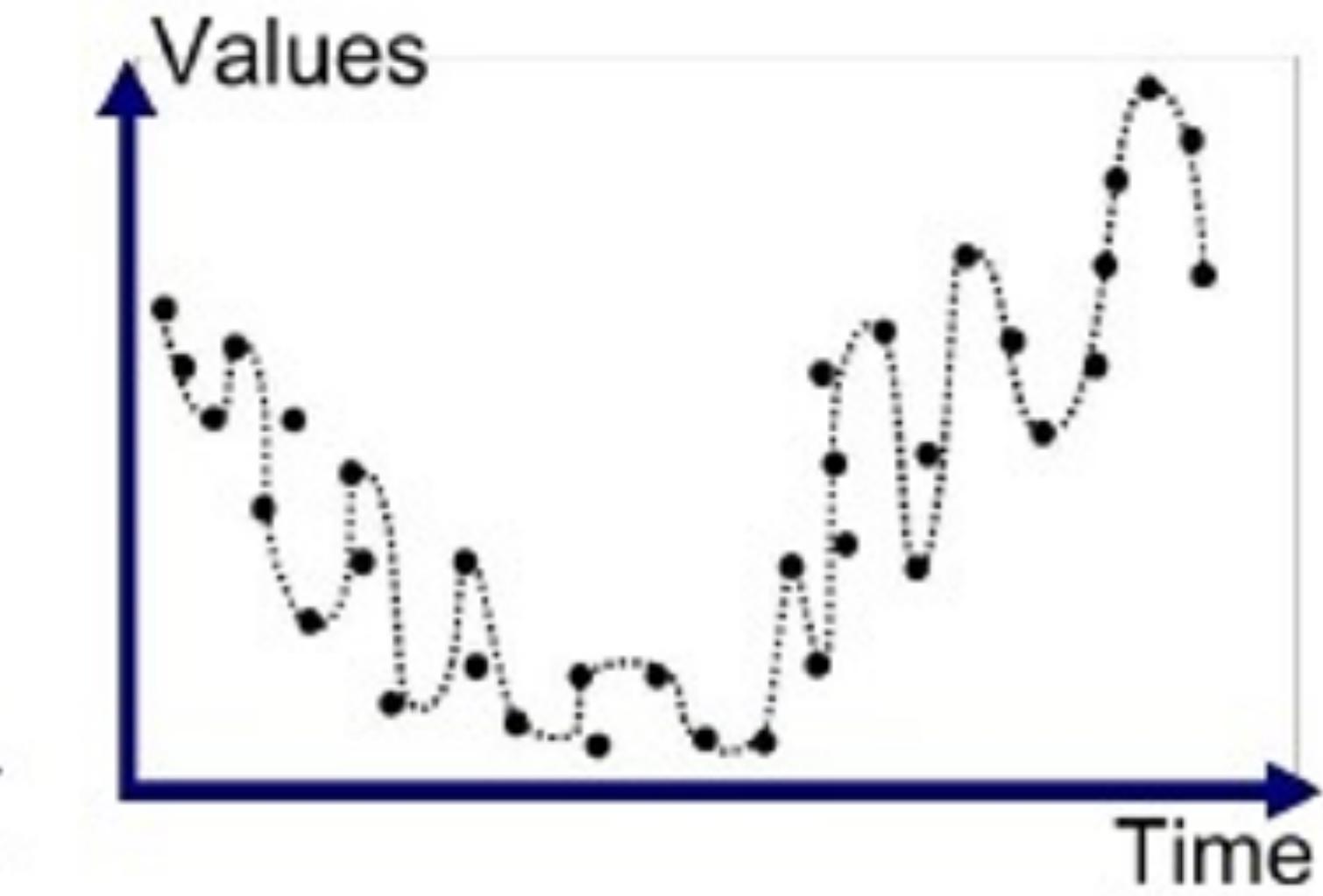
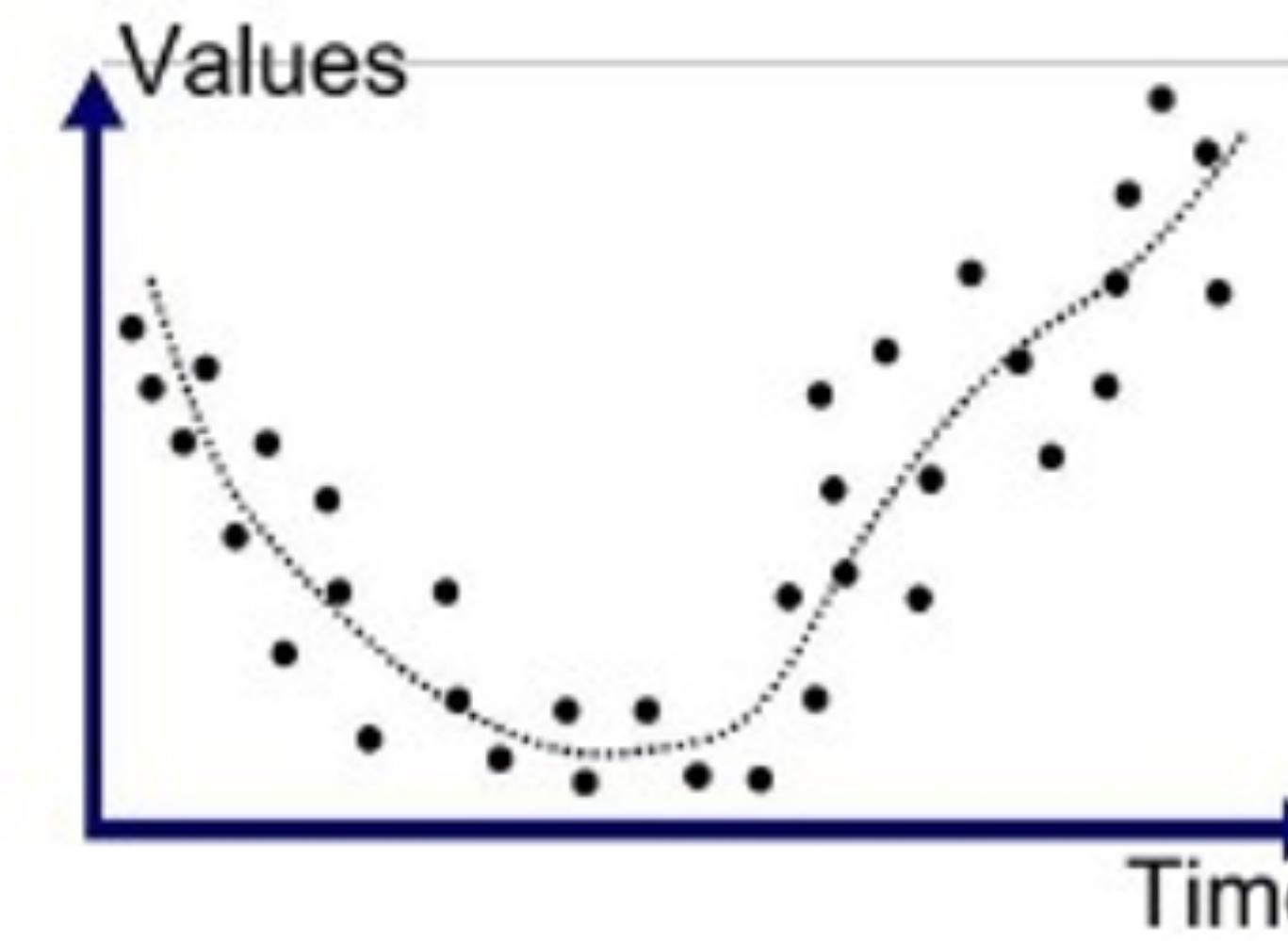
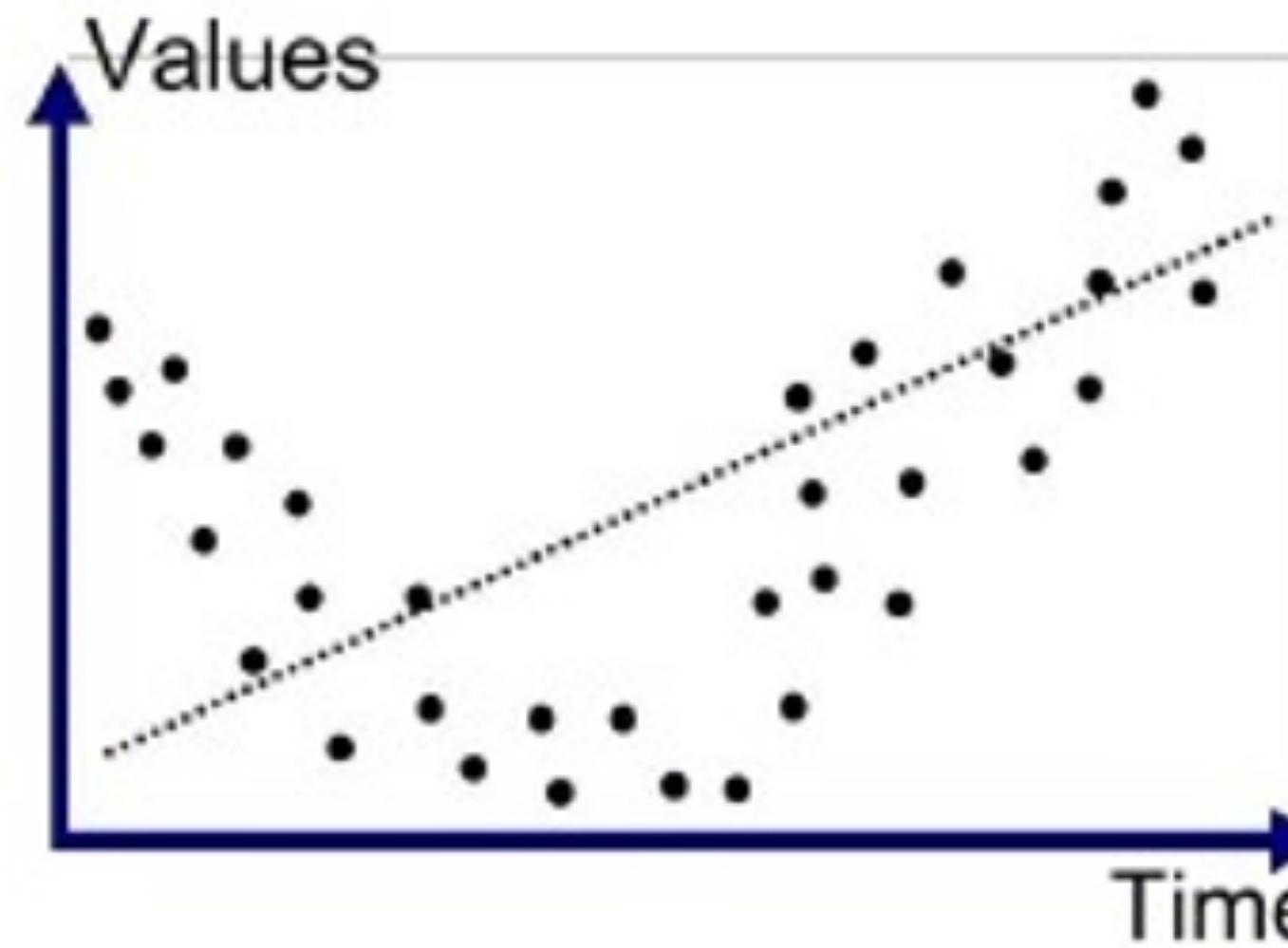


Parameter  $\theta$

Parameter  $\theta$

# The bias-variance tradeoff

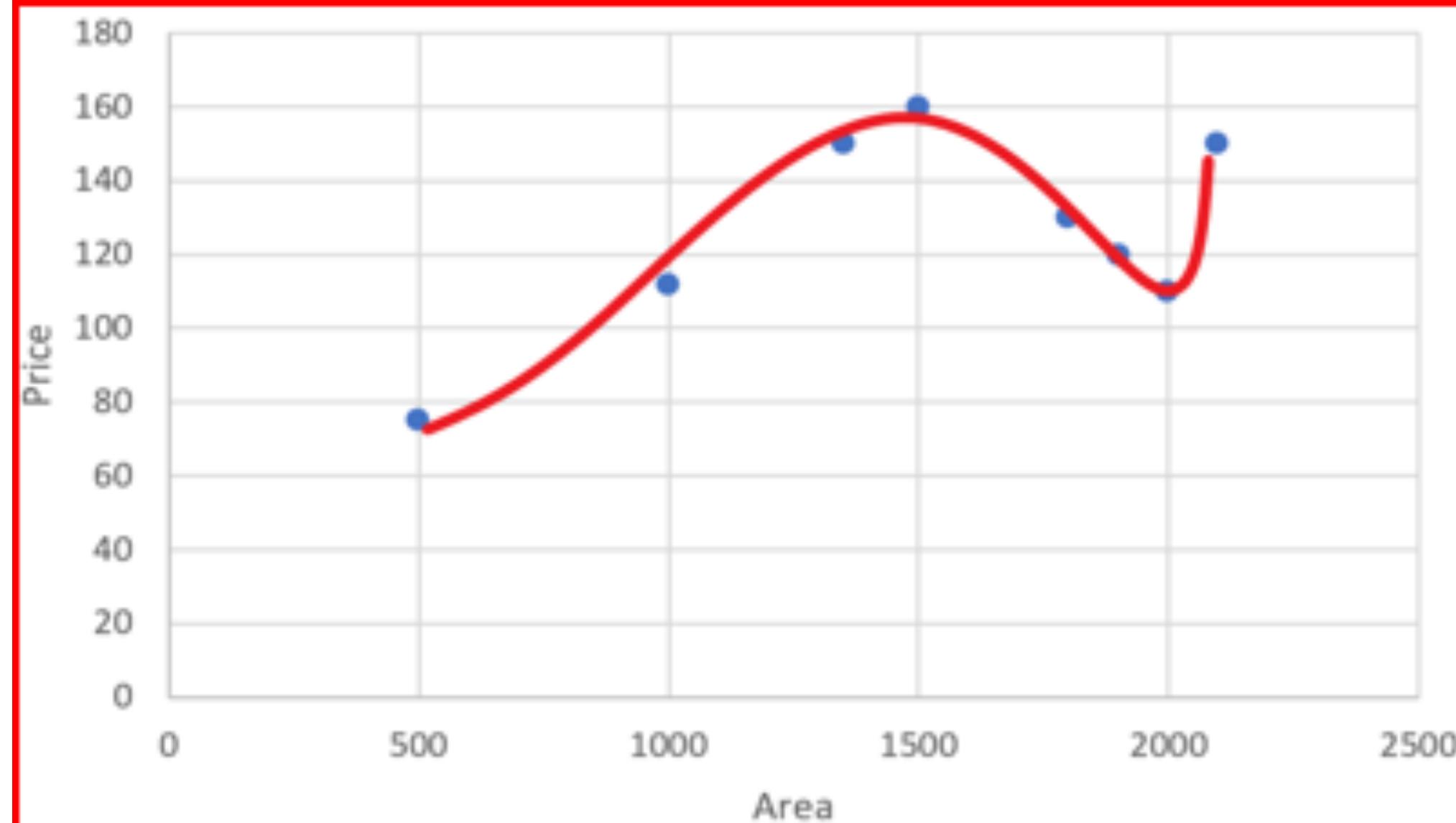
*The lower the losses, the better? Not really.*



# Regularization Techniques

$$L(x, y) = \sum_{i=1}^n (y_i - f(x_i))^2$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$



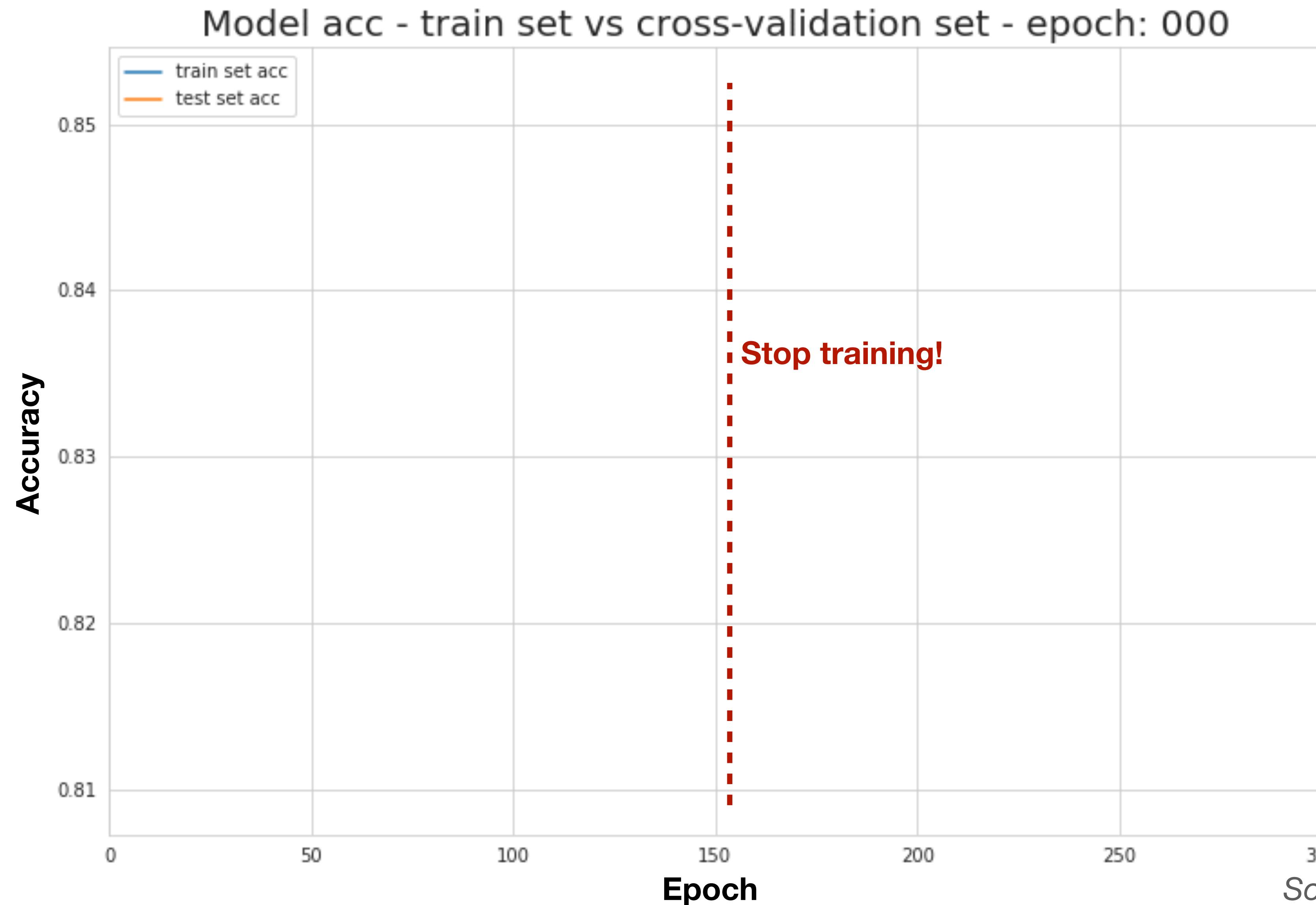
$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$$

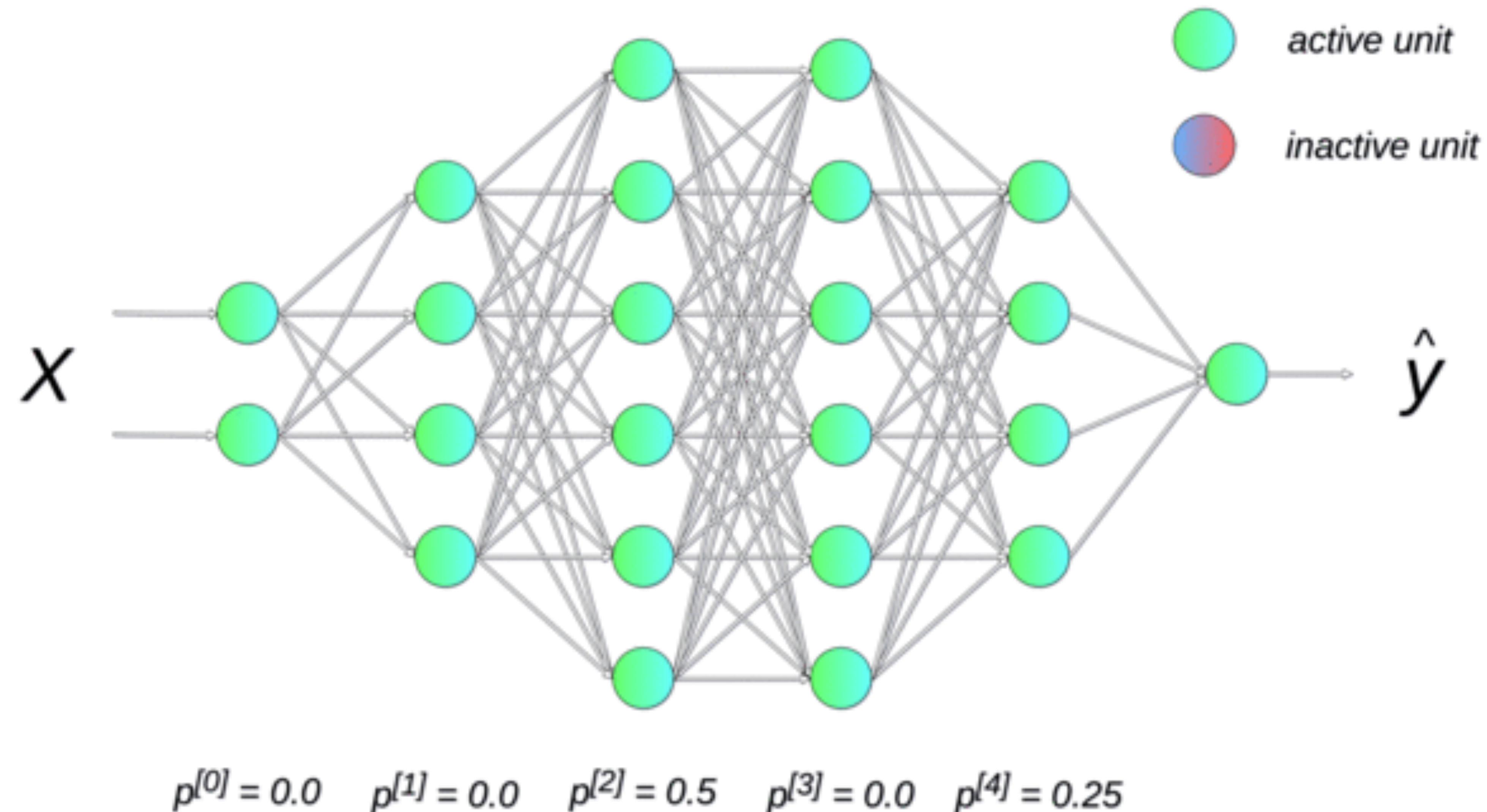
$$L(x, y) \equiv \sum_{i=1}^n (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^n |\theta_i|$$

$$L(x, y) \equiv \sum_{i=1}^n (y_i - h_\theta(x_i))^2 + \lambda \sum_{i=1}^n \theta_i^2$$

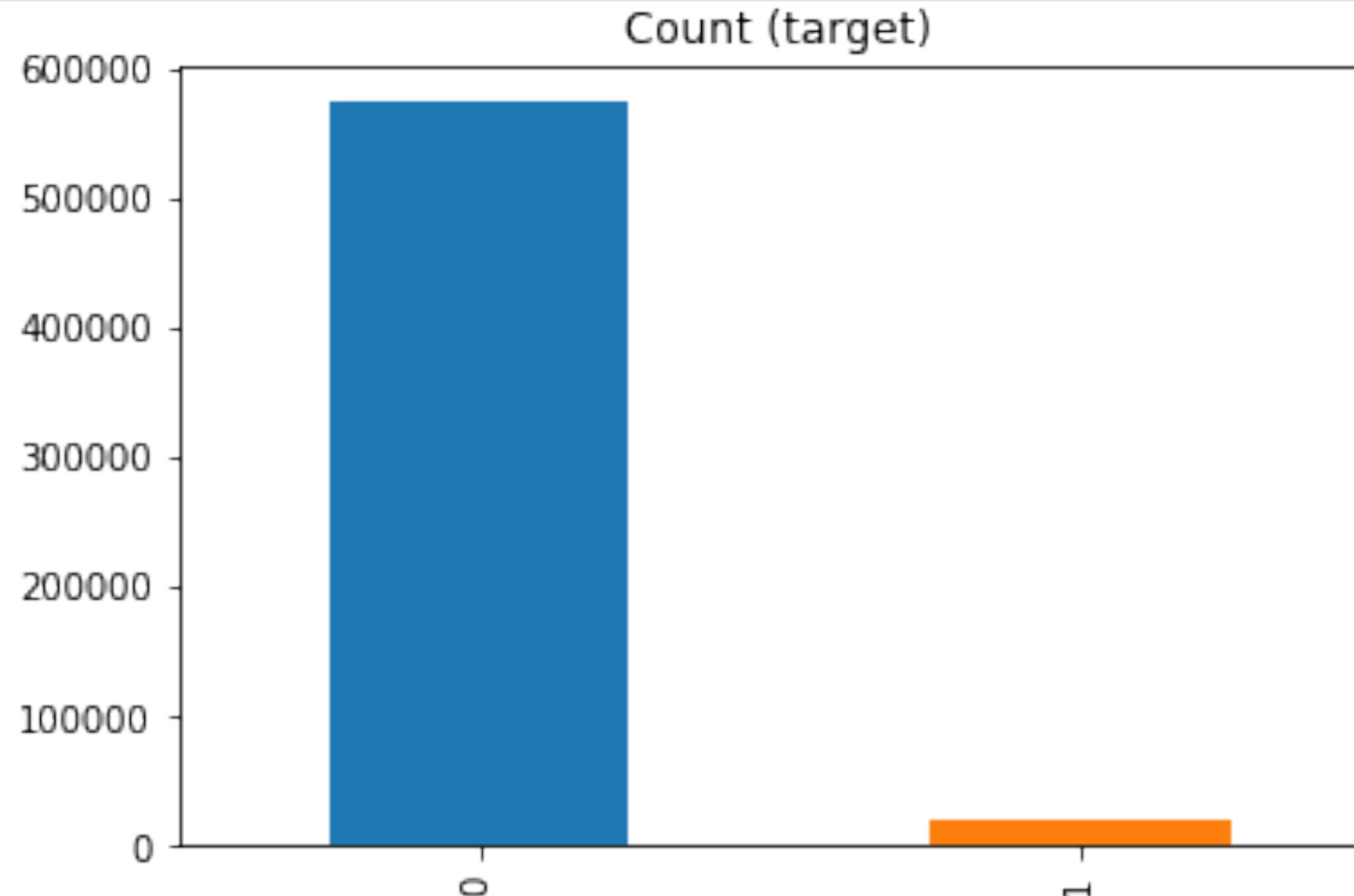
# Early Stopping



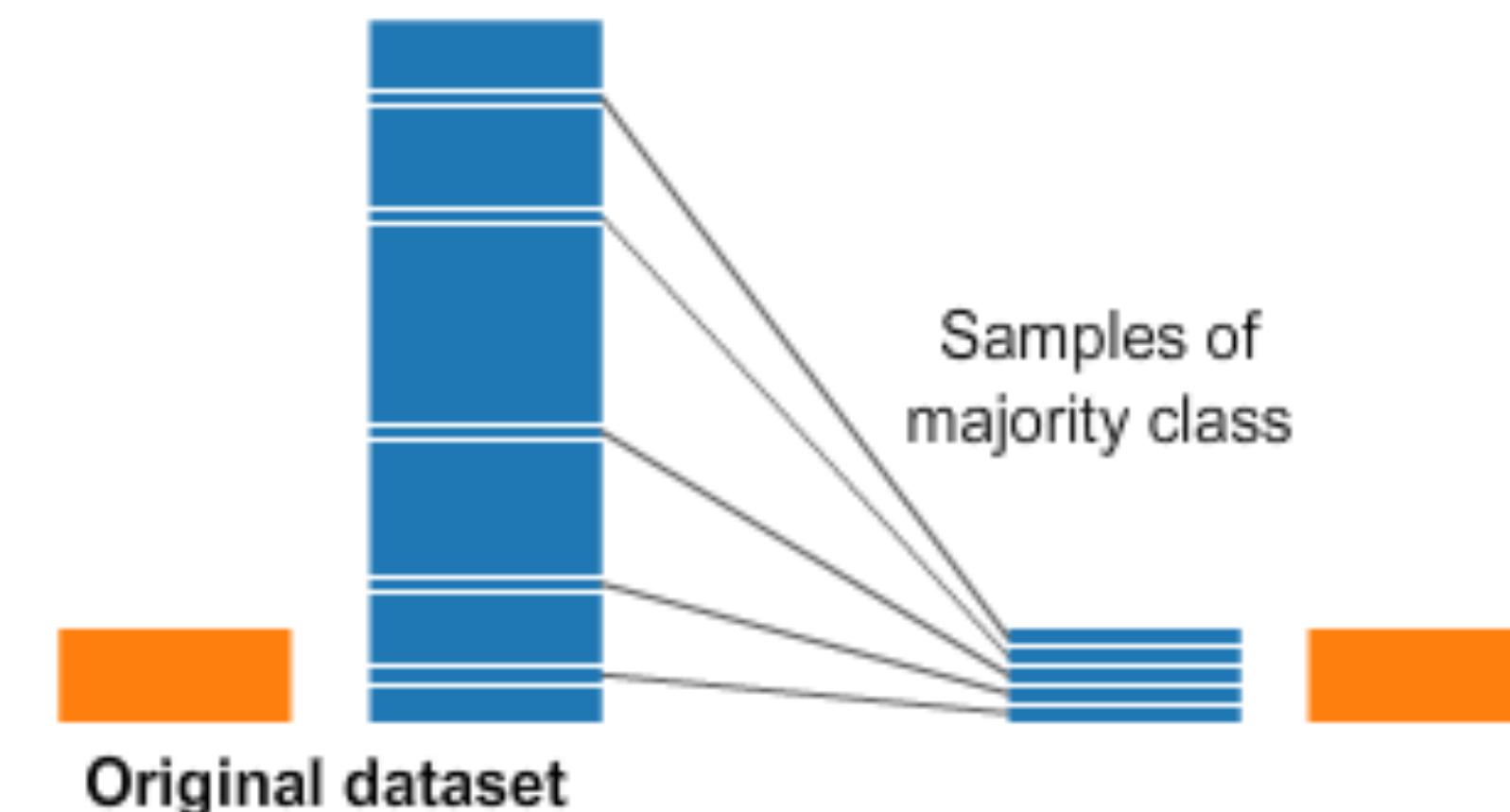
# Dropout



# Balanced/Imbalanced training set



Undersampling

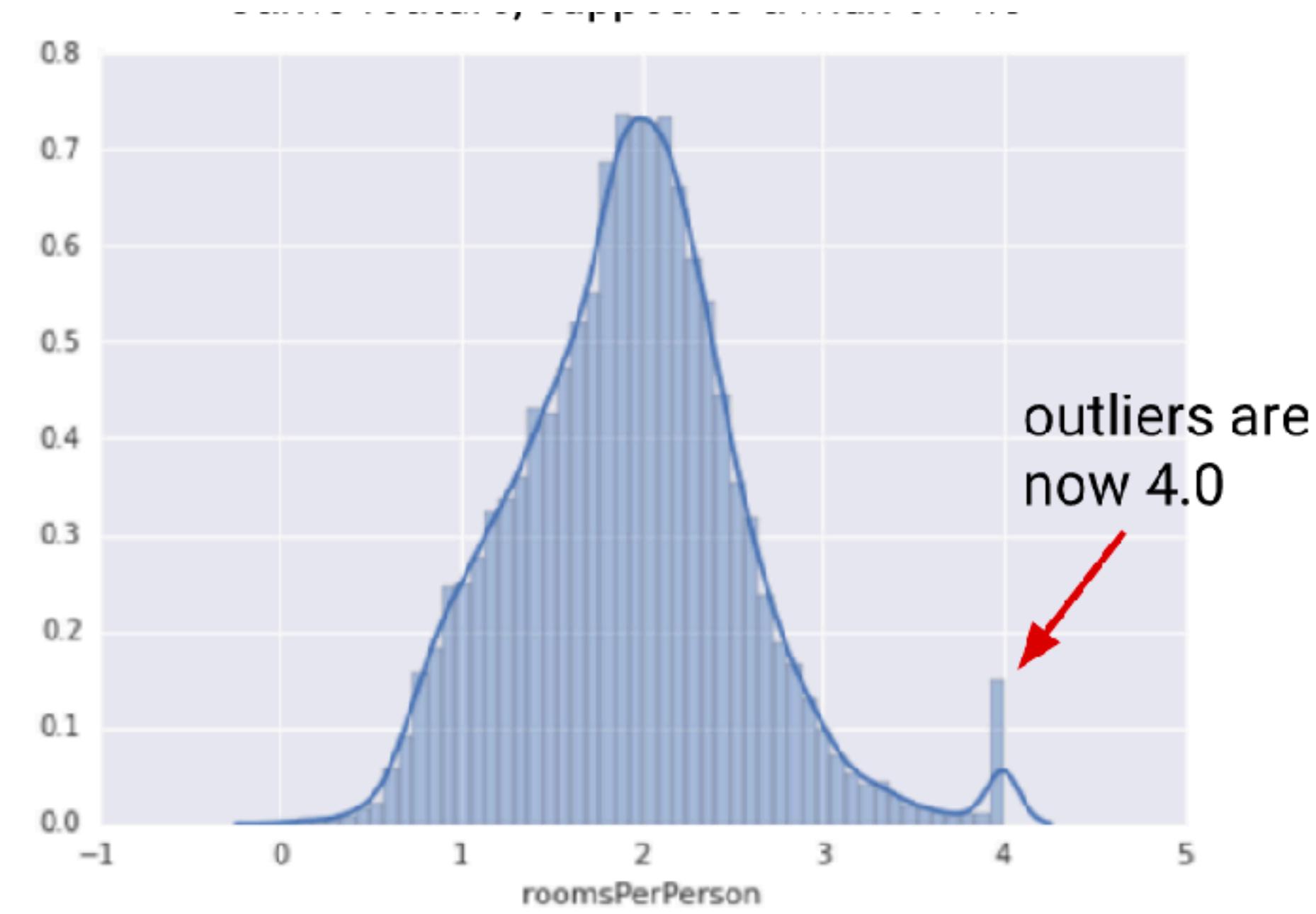
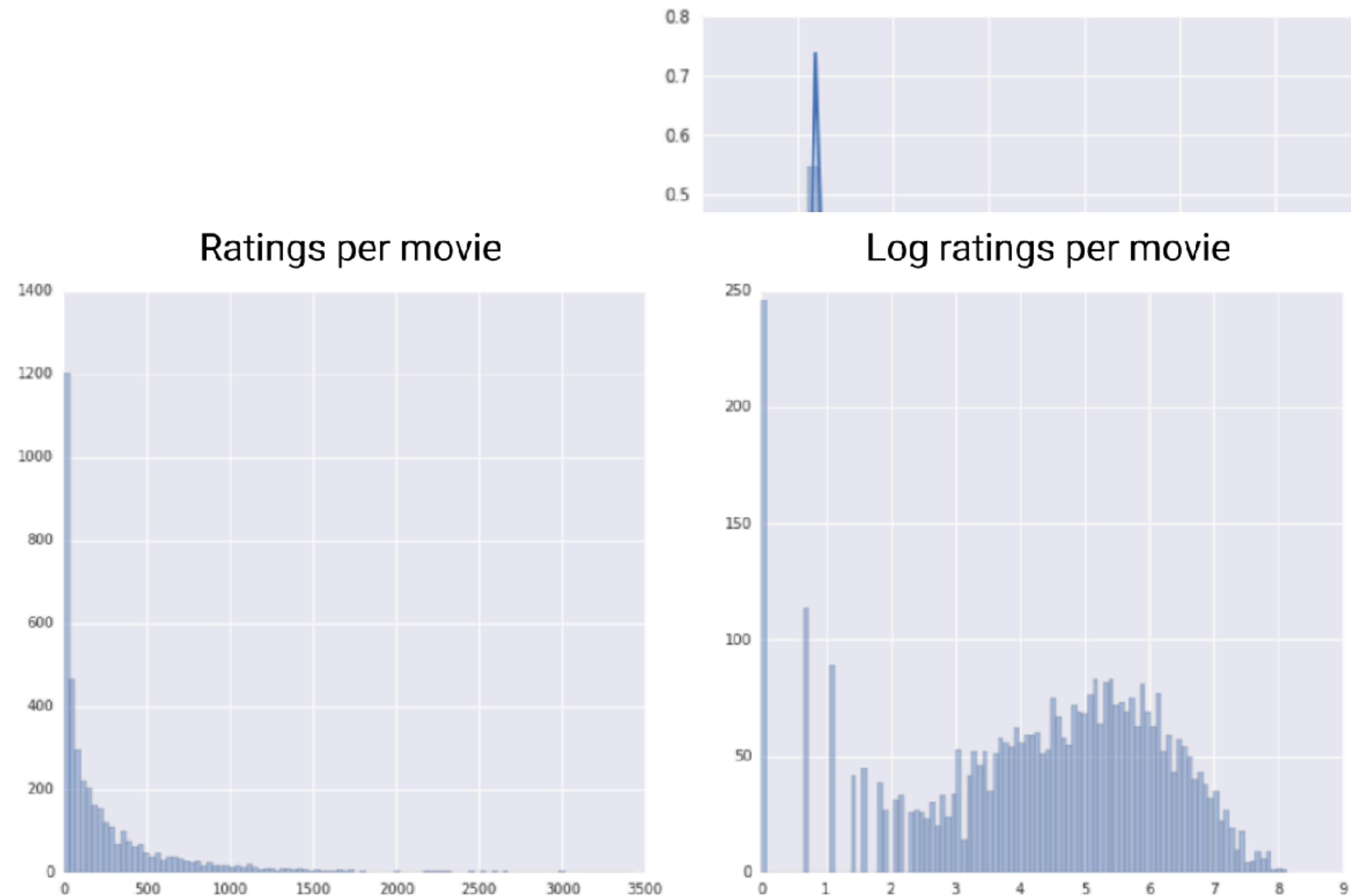


Oversampling



# Data Normalization

A process to transform the input **data** in a **well-behaved** form.

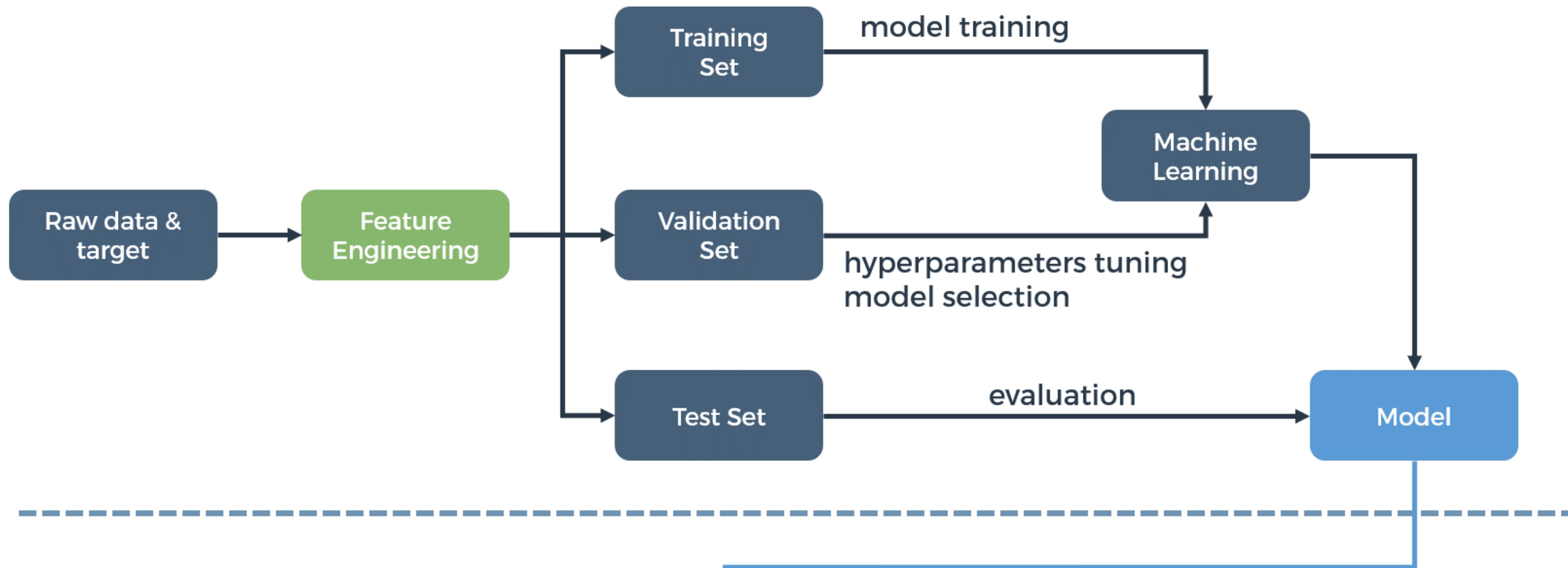


Further reading: [sklearn scalers](#)

Source: [developers.google.com](http://developers.google.com)

# General Workflow of ML/DL

## TRAINING



# Categories of machine learning

## Supervised

### Classification

- Naive bayes
- Support vector machine
- Decision tree
- Random forest
- K-Nearest Neighbor
- Logistic regression

### Regression

- Linear
- Generalized

## Unsupervised

### Clustering

- K-means
- K-medoids

### Dimensionality reduction

- PCA
- SVD

## Reinforcement

### Discrete

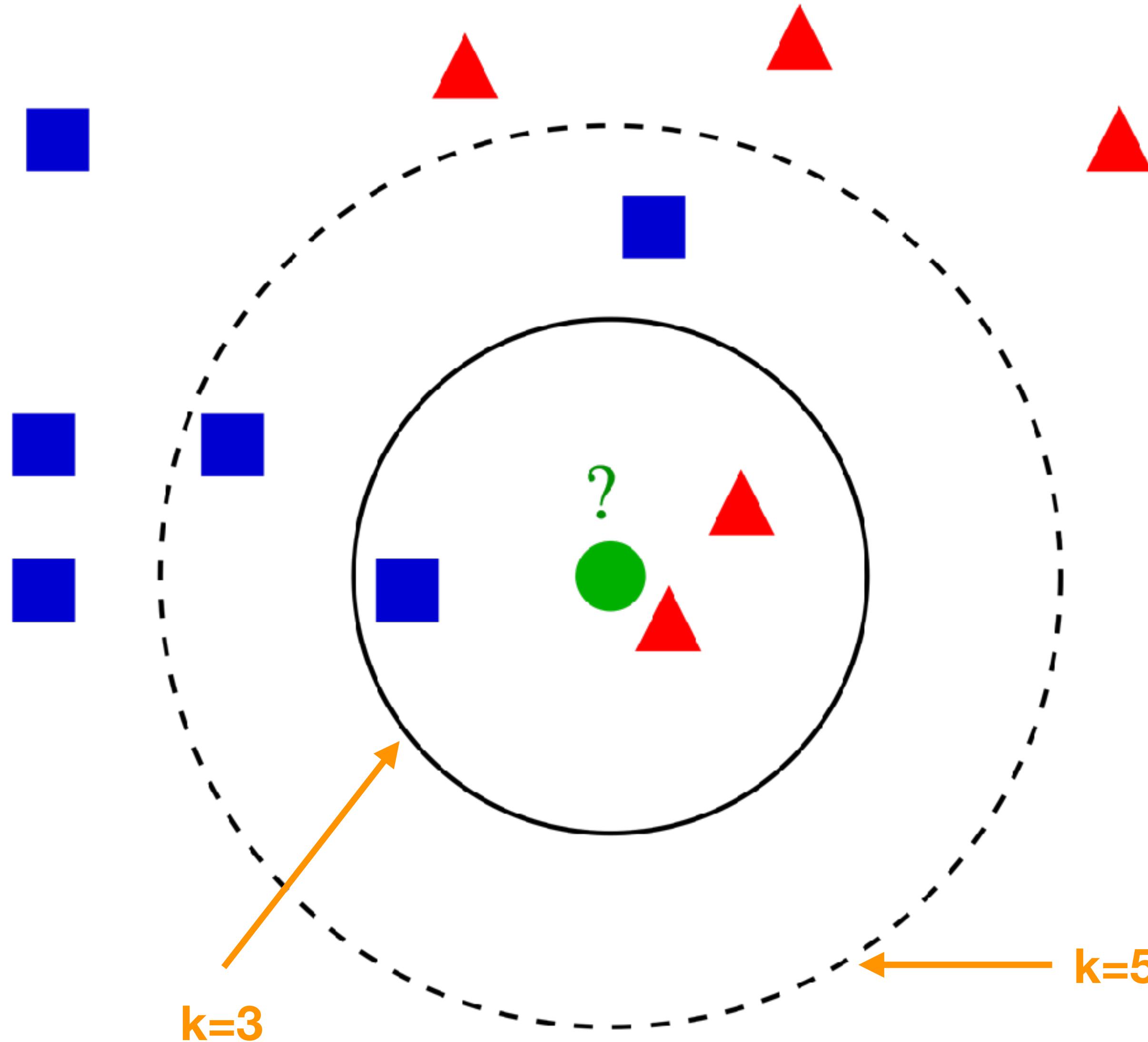
- Markov decision process
- Deep Q Network
- A2C
- A3C

### Continuous

- DDPG
- NAF

*... and many more... This list is incomplete!*

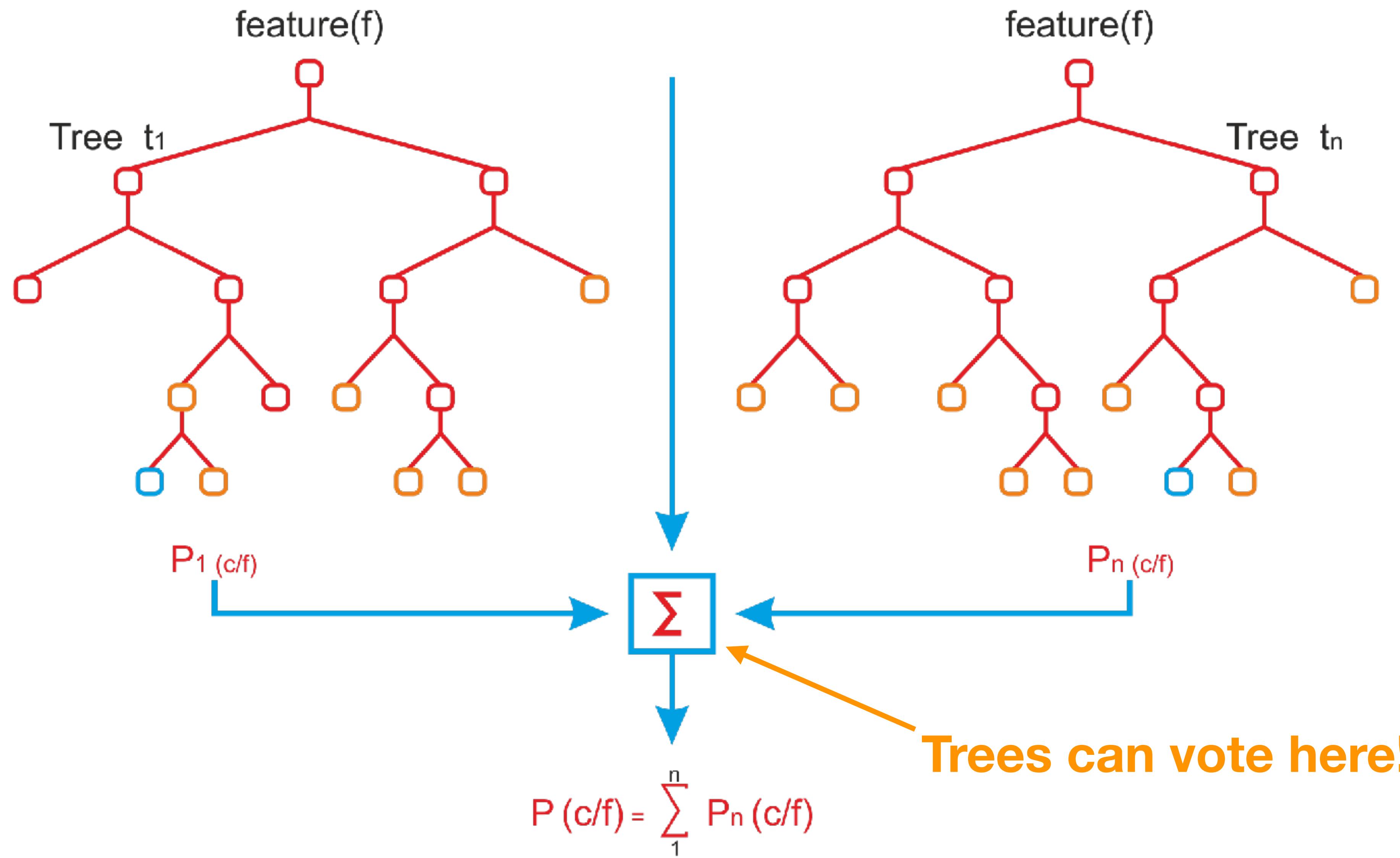
# Classification: $k$ -Nearest Neighbor



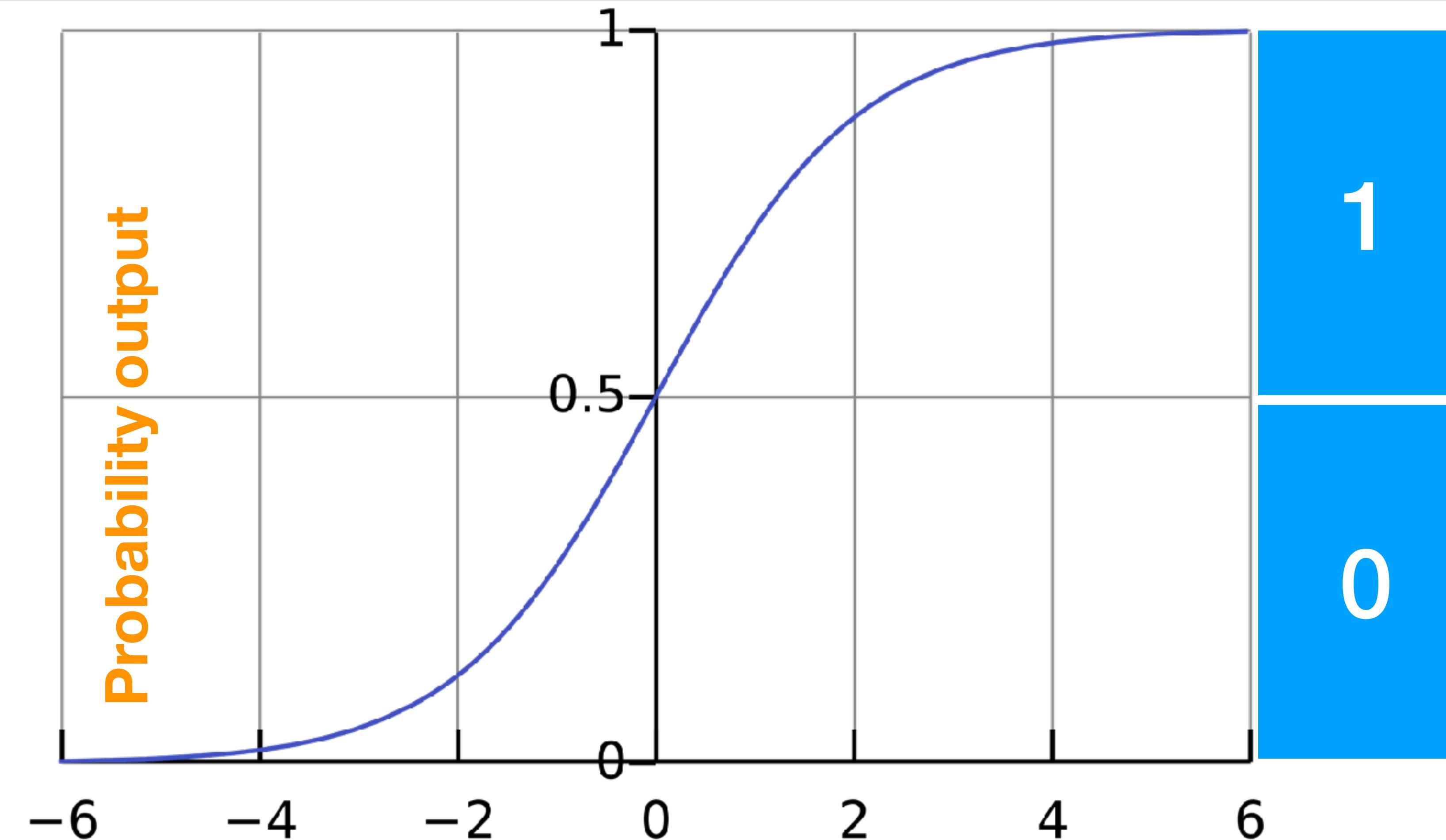
... is a **classification** algorithm

... based on a **voting** process

# Classification: Decision Tree → Random Forest



# Classification: Logistic Regression



**Logistic sigmoid function**

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

... is a binary **classification** algorithm, not a regression algorithm  
... Gives the **probability** of each class

# Confusion Matrix

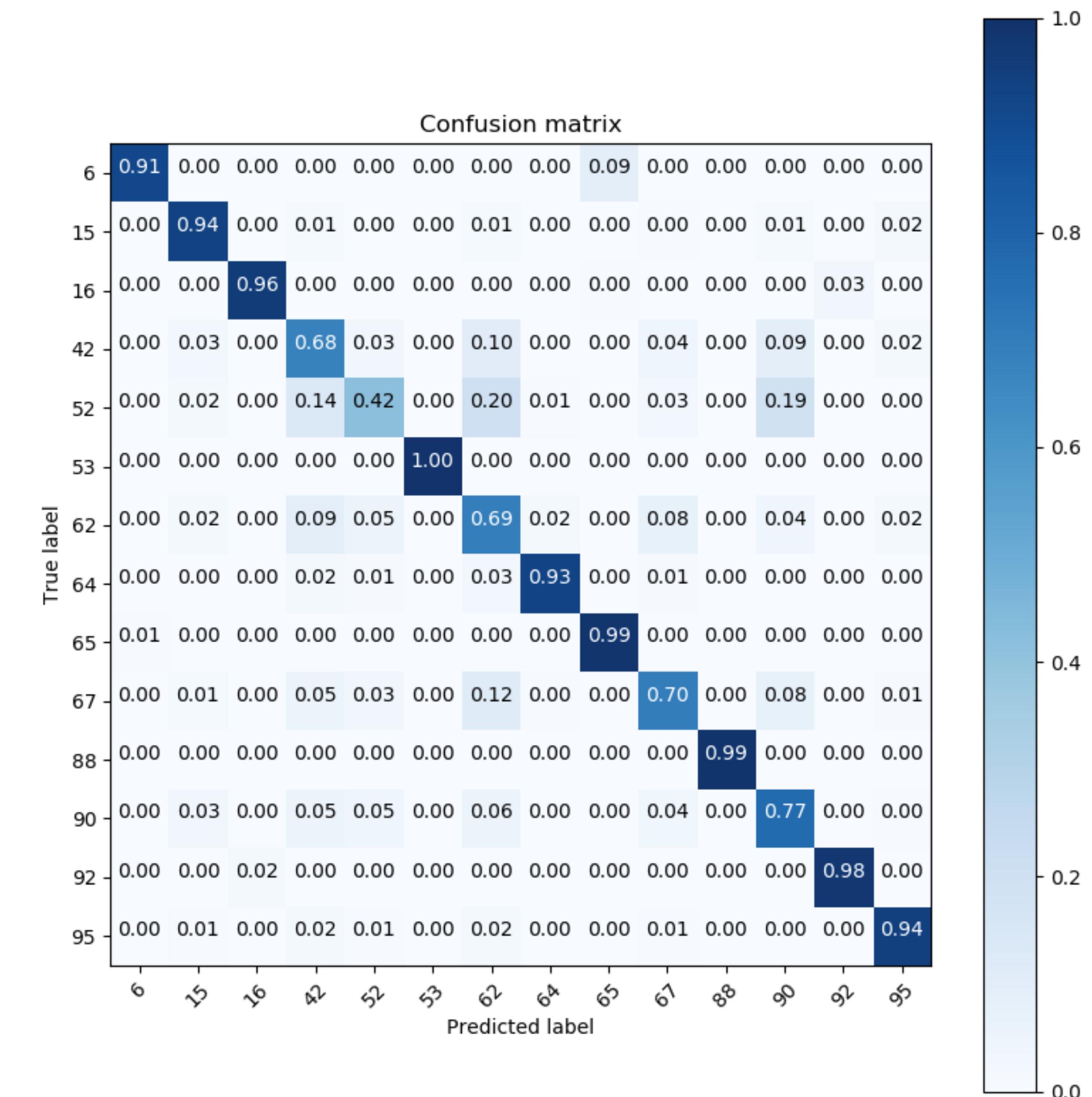
		Prediction outcome		tot
		P	n	
actual value	p'	True Positive	False Negative	P'
	n'	False Positive	True Negative	N'
total		P	N	

**Accuracy =  $(TP + TN) / (TP + FN + FP + TN)$**

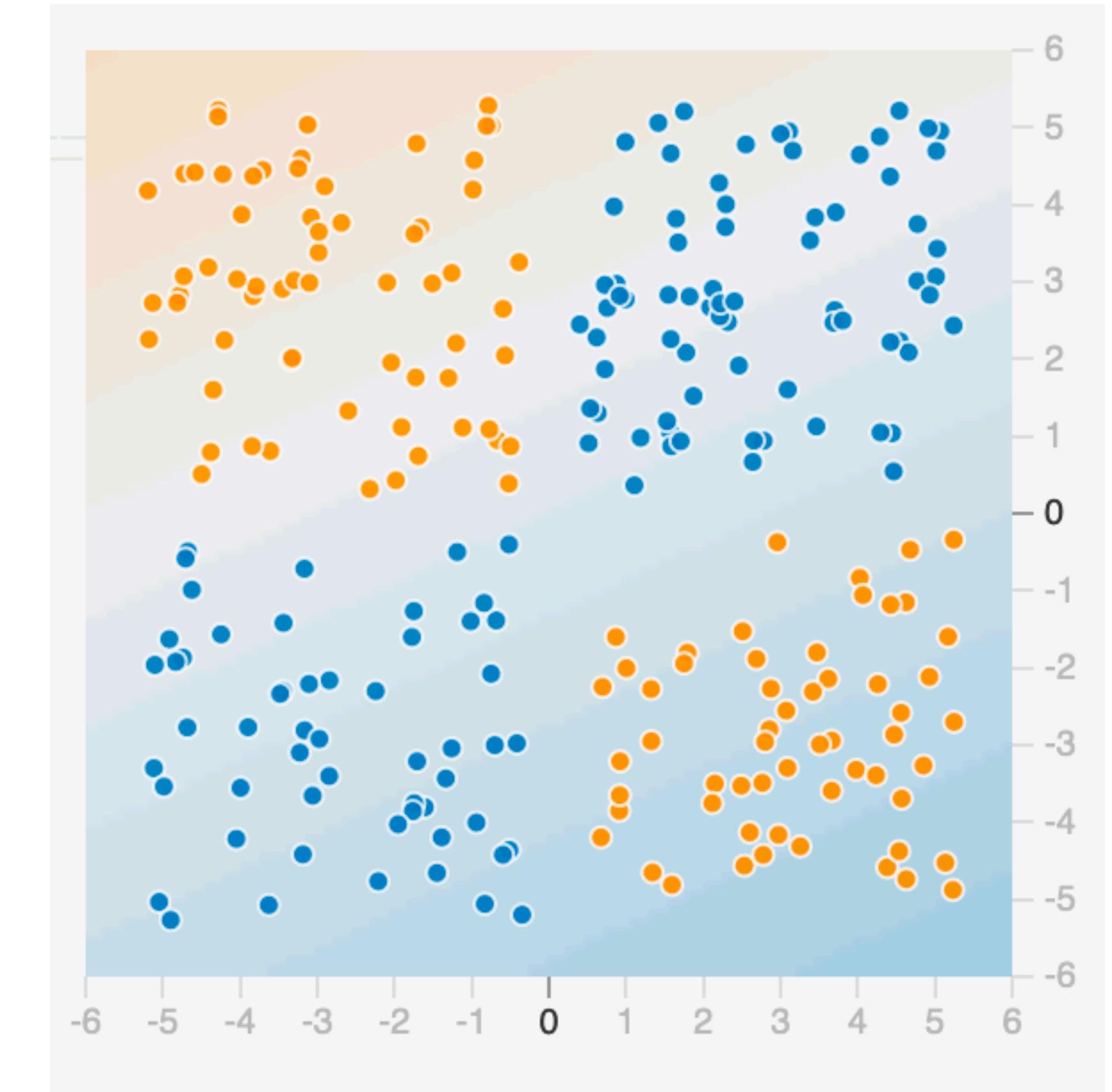
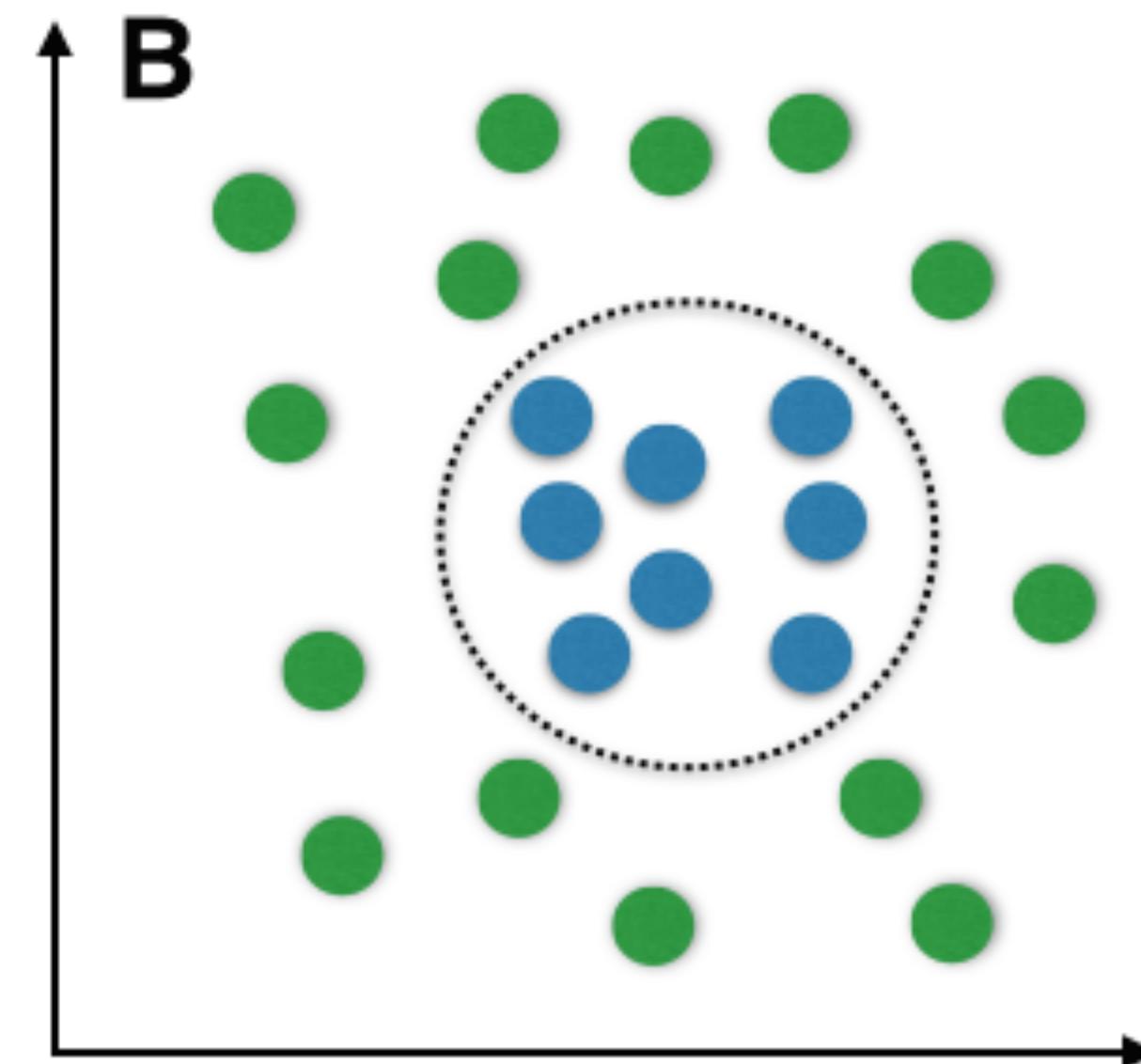
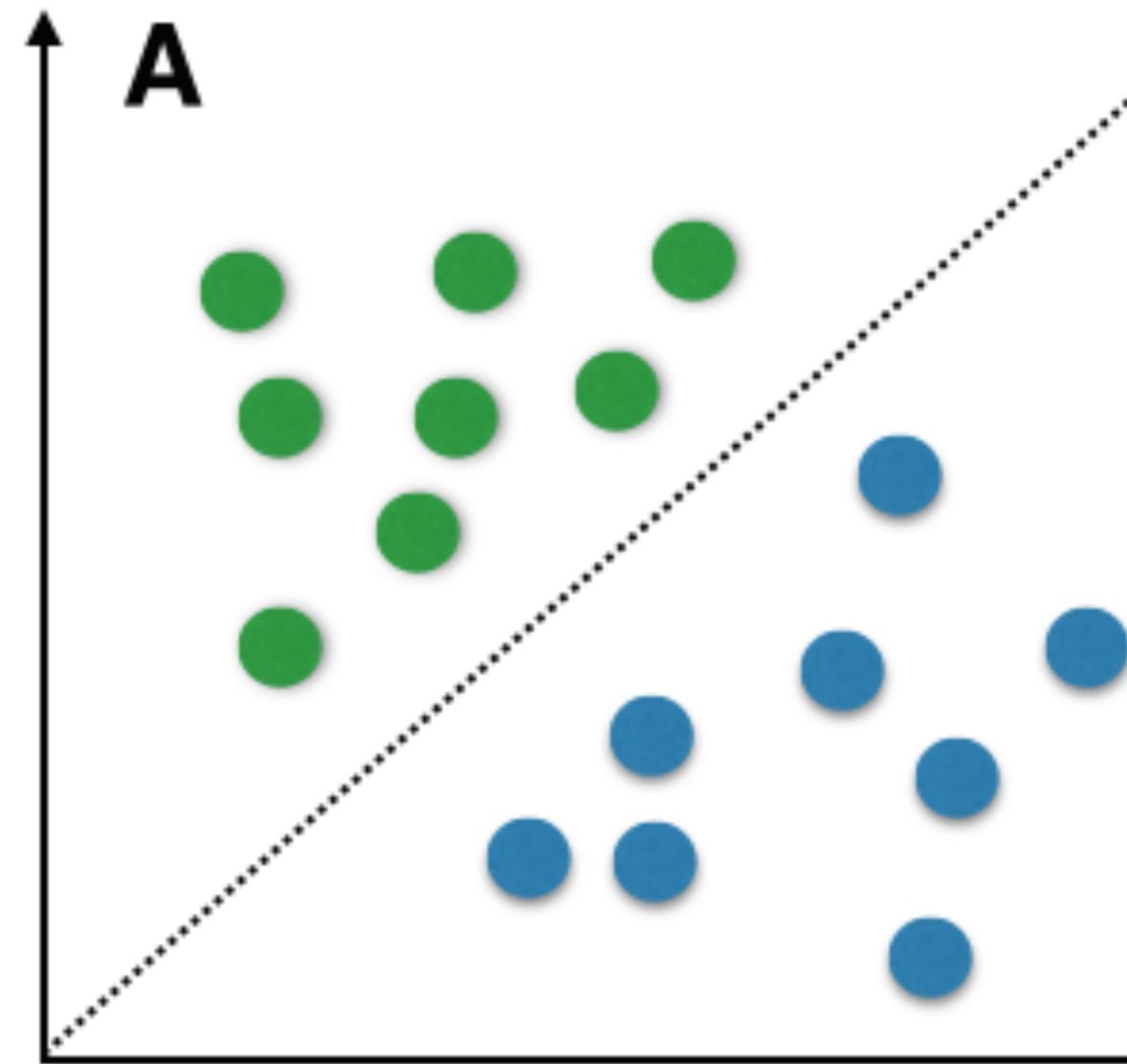
**Precision (p) = TP / (TP + FP)**

**Recall (r) = TP / (TP + FN)**

$$F_1 = \frac{2}{r^{-1} + p^{-1}}$$



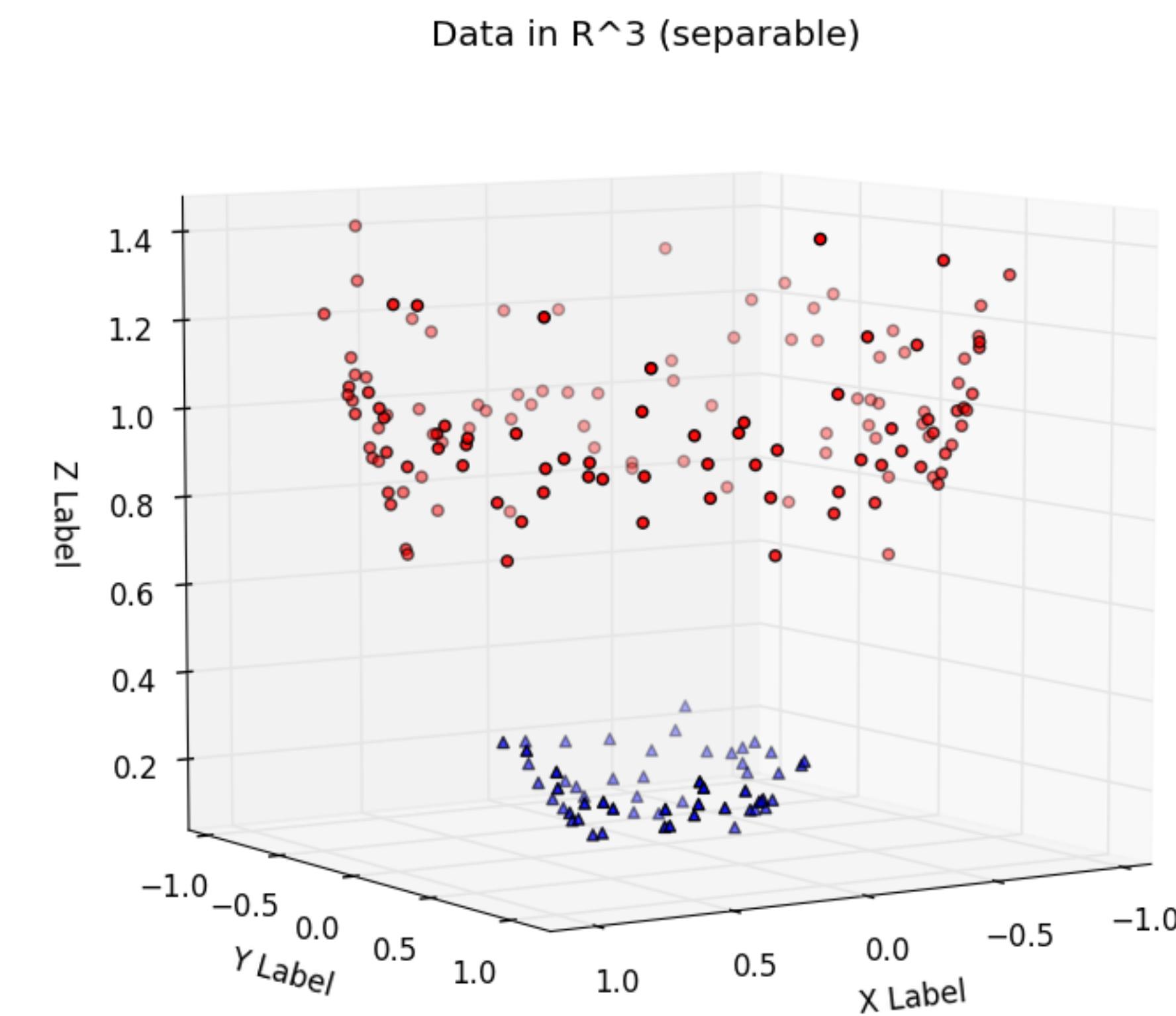
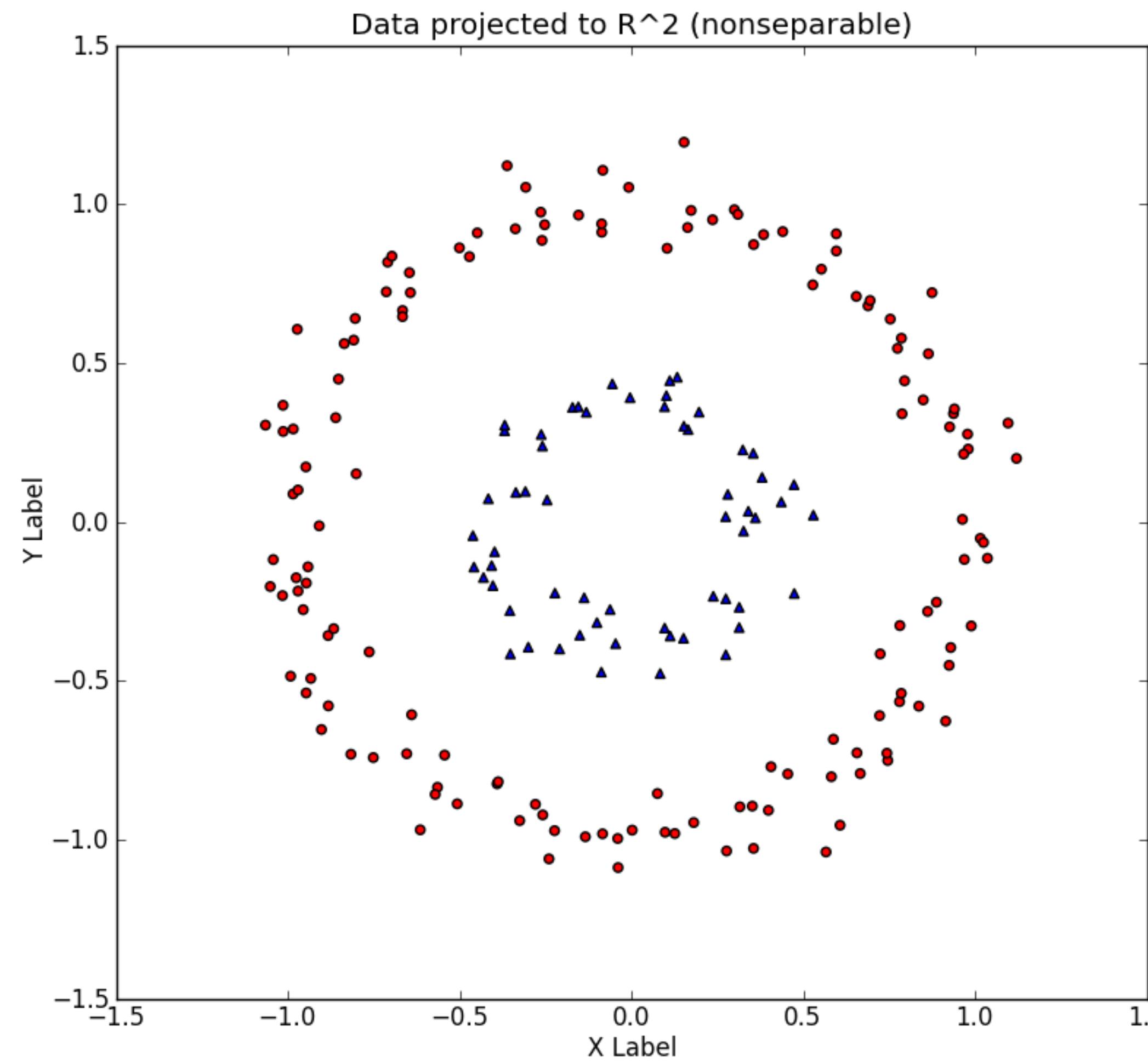
# What if the problem is not so simple?



## Possible solutions:

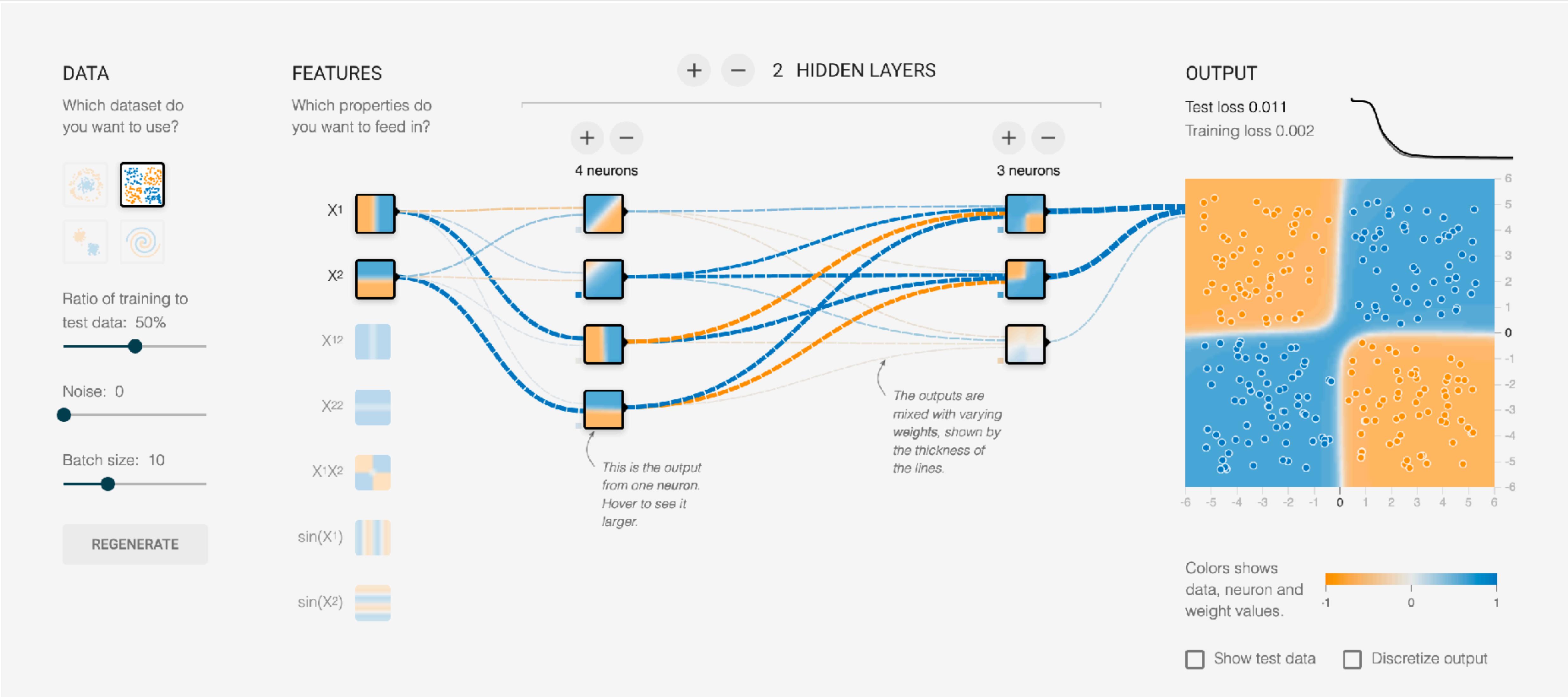
- map into another space (kernel trick)
- Add more layers (deep learning)

# Kernel Trick



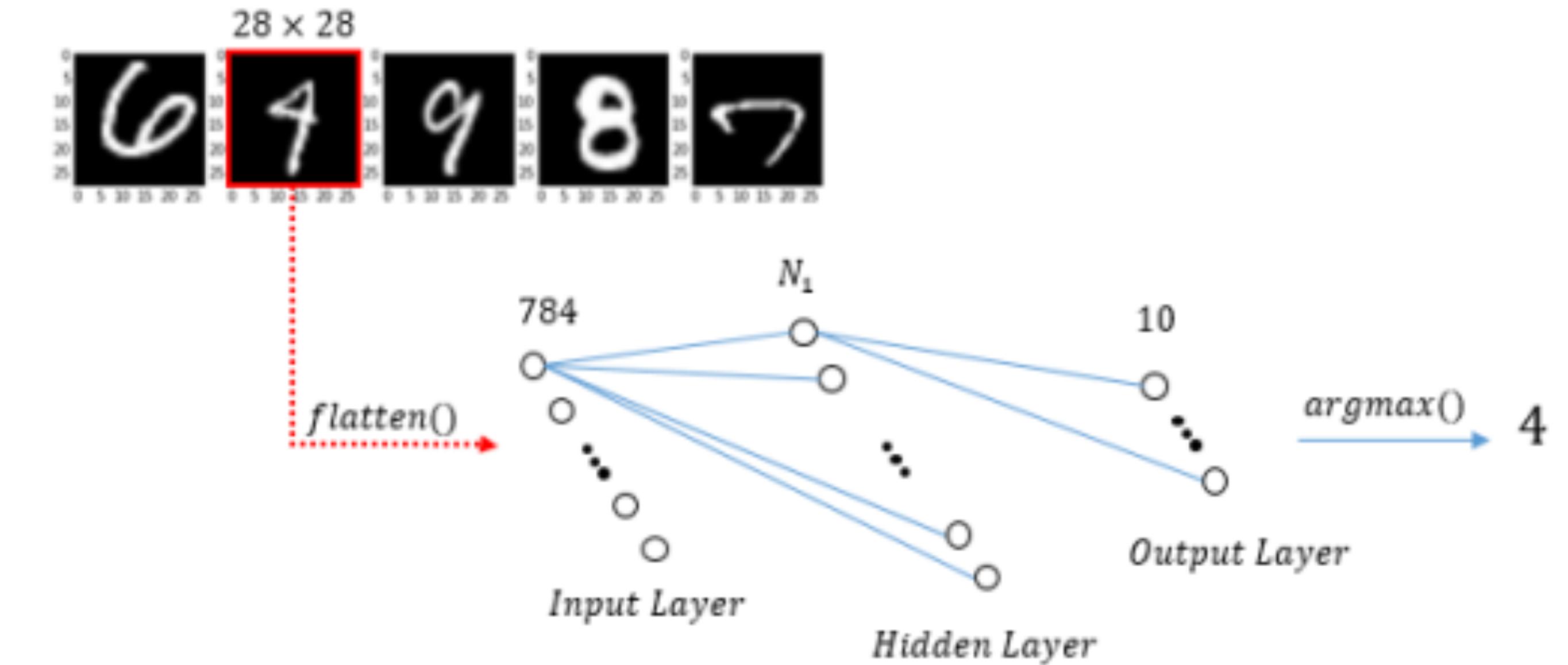
**Or, use a deeper neural network...**

# Live demo: Multi-layer Perceptrons



# Multi-Layer Perception

3 4 2 1 9 5 6 2 1 8  
8 9 1 2 5 0 0 6 6 4  
6 7 0 1 6 3 6 3 7 0  
3 7 7 9 4 6 6 1 8 2  
2 9 3 4 3 9 8 7 2 5  
1 5 9 8 3 6 5 7 2 3  
9 3 1 9 1 5 8 0 8 4  
5 6 2 6 8 5 8 8 9 9  
3 7 7 0 9 4 8 5 4 3  
7 9 6 4 1 0 4 9 2 3



# Deep Neural Network ==> Deep Learning

What about these images? Should we make the neural network as deep as possible?

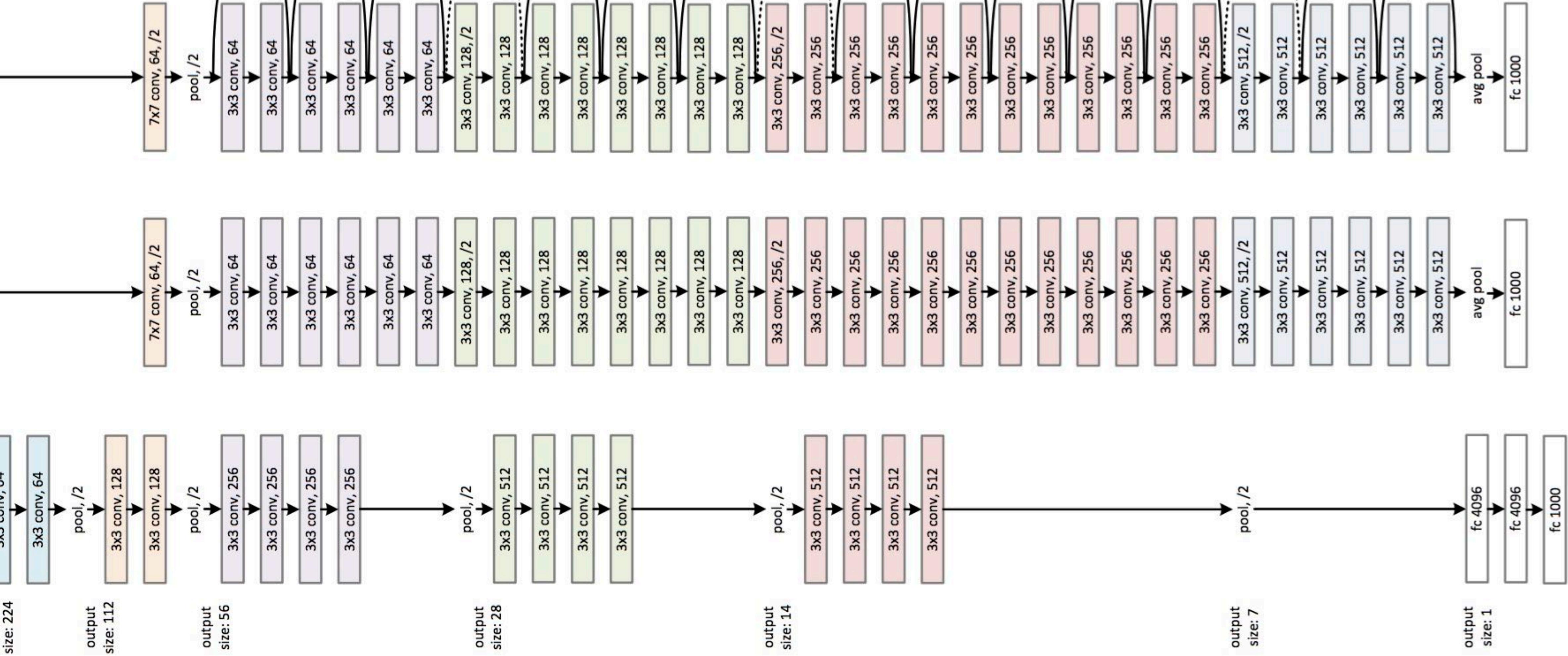
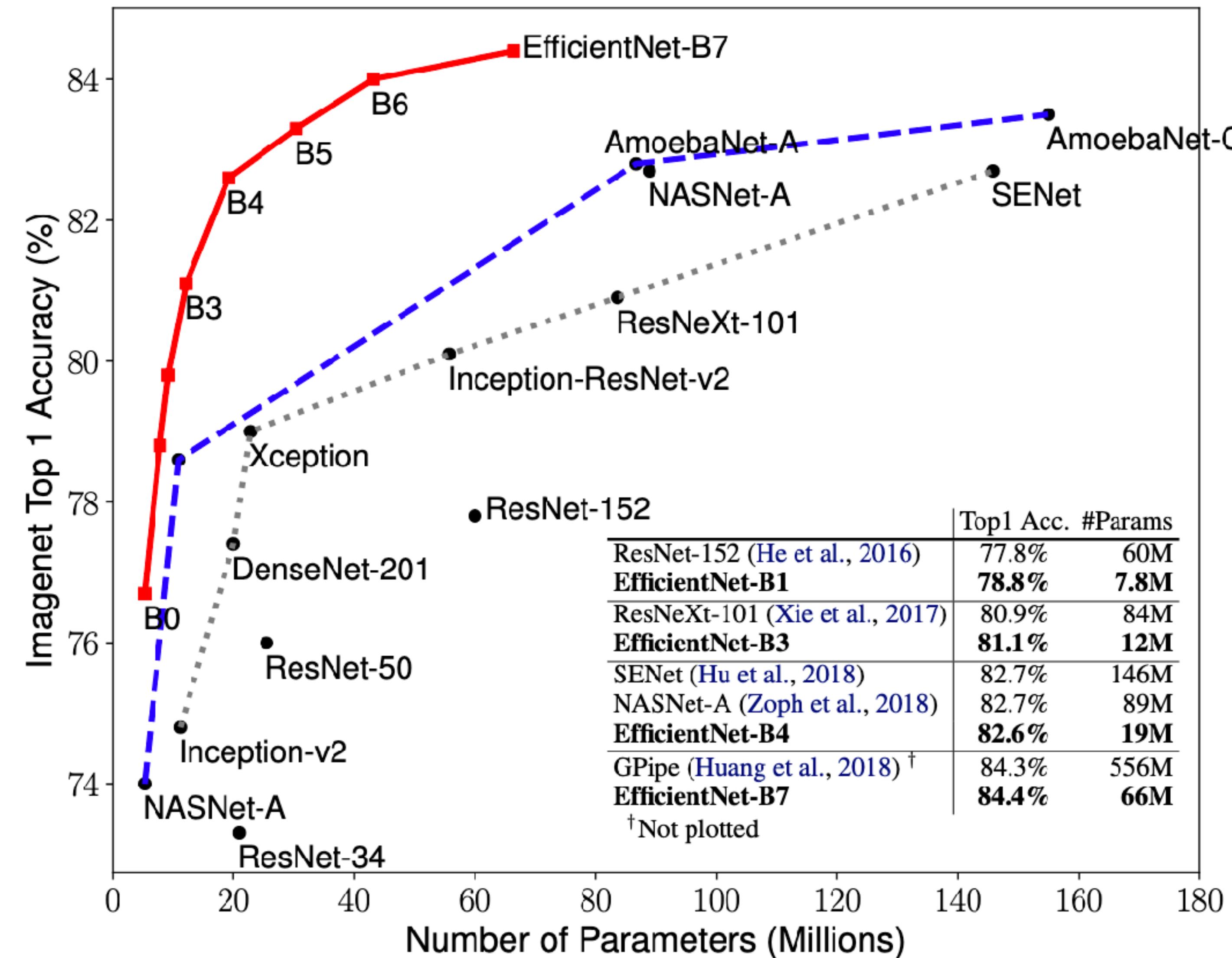


Figure 3. Example network architectures for ImageNet. Left: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. Middle: a plain network with 34 parameter layers (3.6 billion FLOPs).

# Deep Neural Network ==> Deep Learning

Tan & Le (2019, arXiv:1905.11946)



# CNNs: Building Blocks of Computer Vision

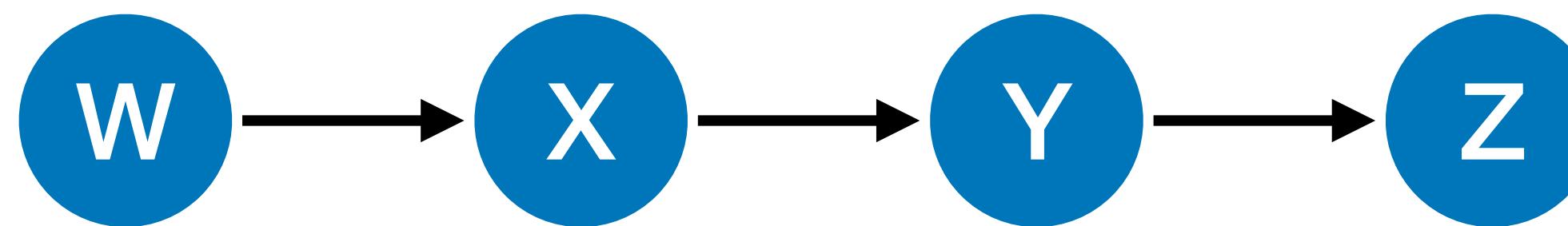


Source: Priya Dwivedi/Medium

# Neural Network: the deeper, the better?

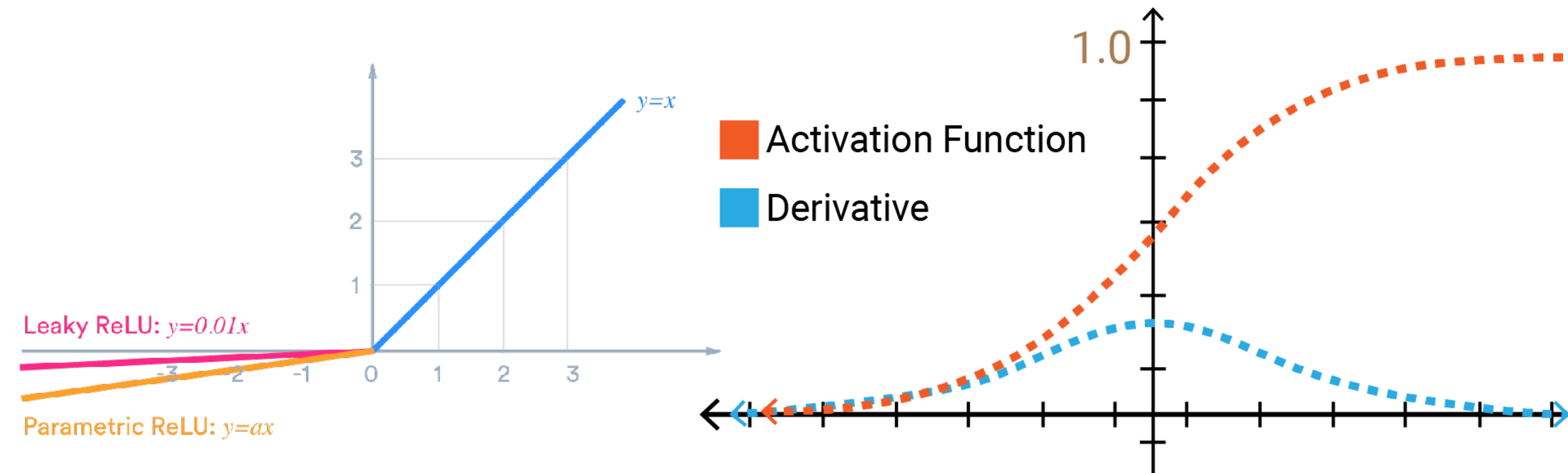
*Not really.*

# The vanishing gradient problem



**Chain rule**

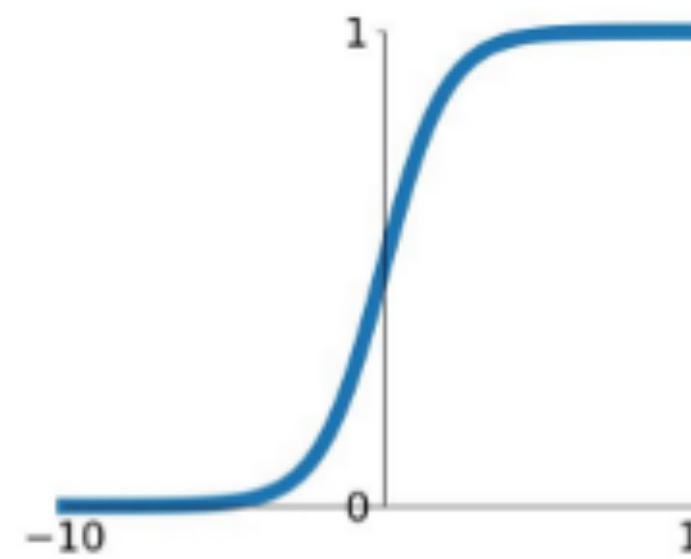
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} \frac{\partial x}{\partial w}$$



# Activation Functions

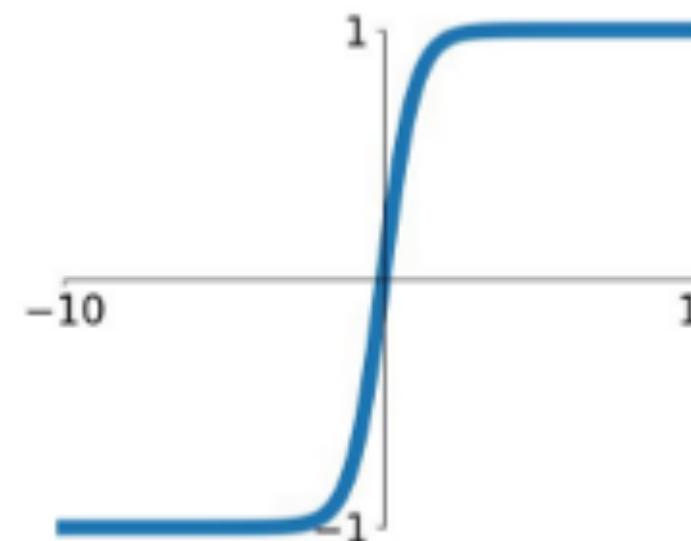
**Sigmoid**

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



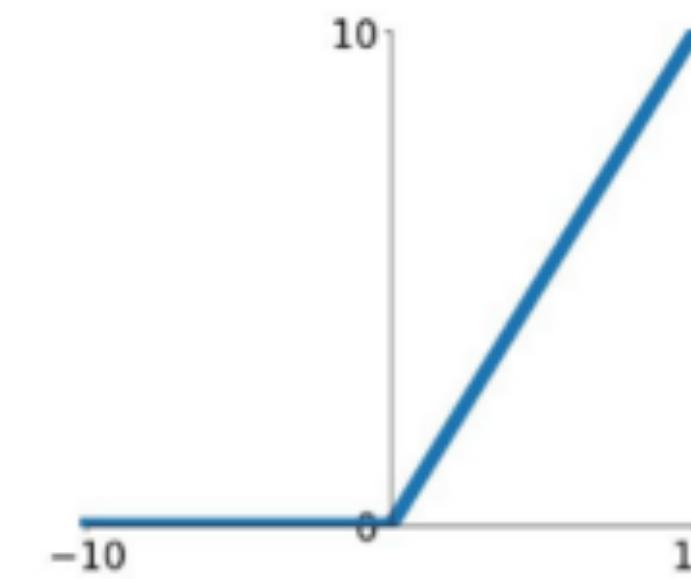
**tanh**

$$\tanh(x)$$



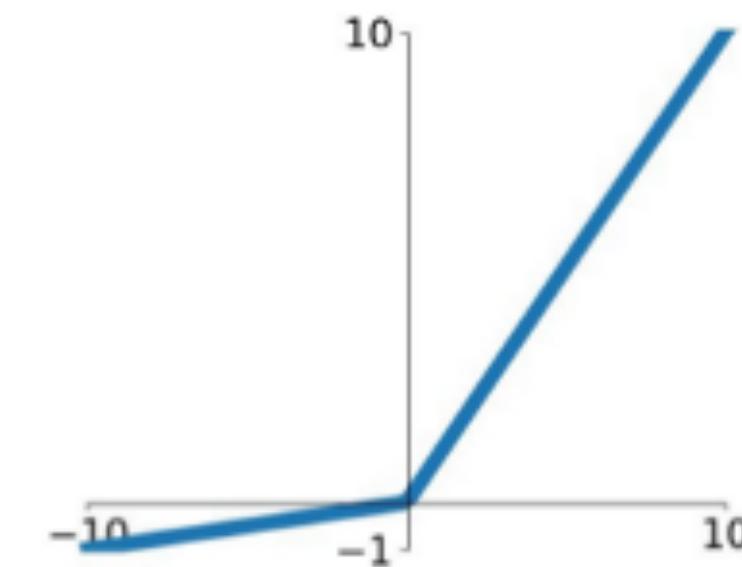
**ReLU**

$$\max(0, x)$$



**Leaky ReLU**

$$\max(0.1x, x)$$

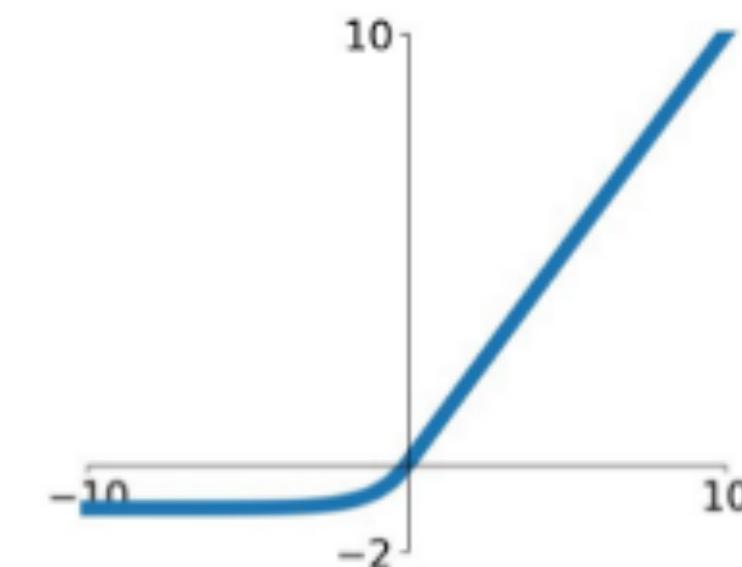


**Maxout**

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

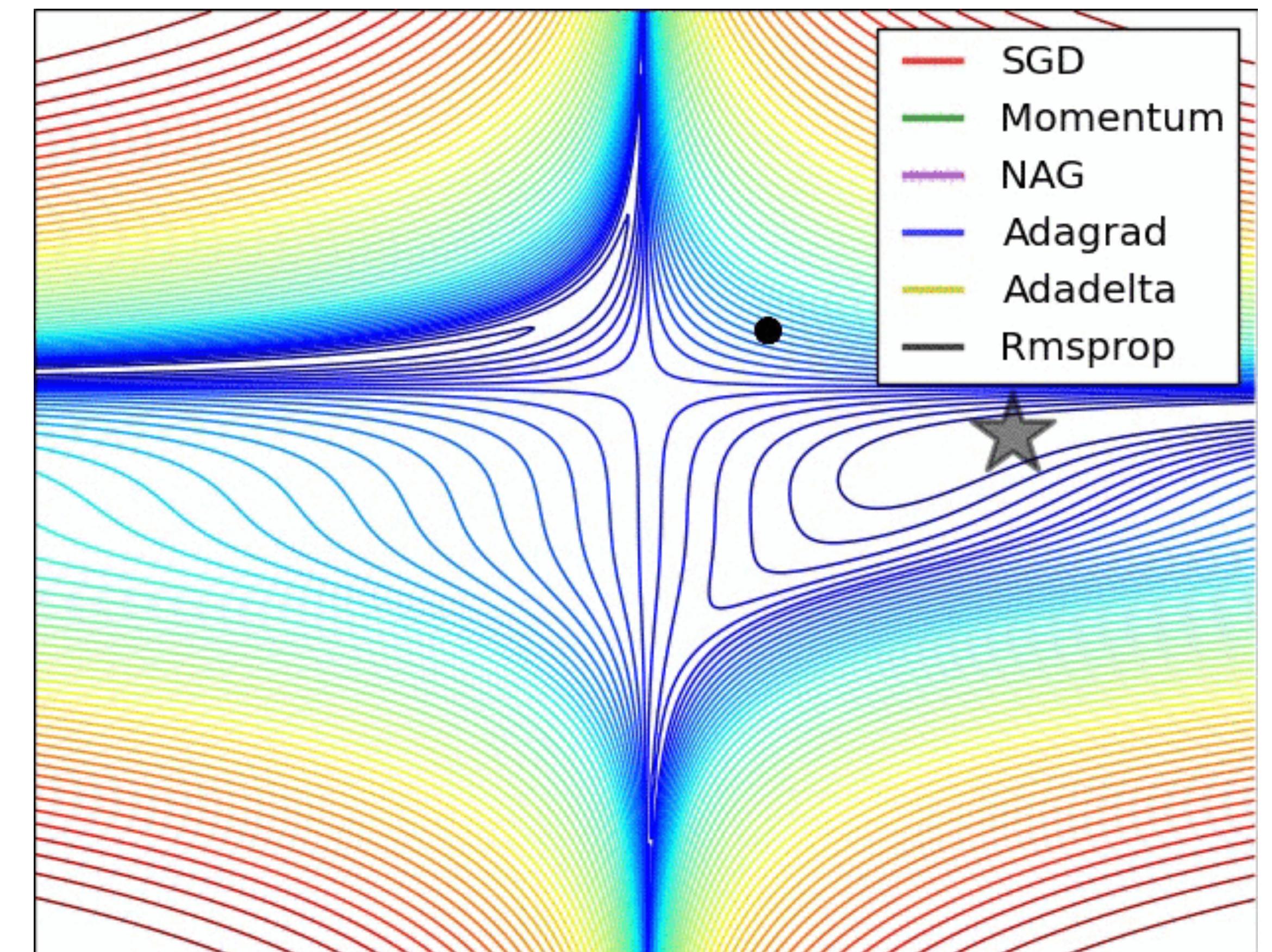
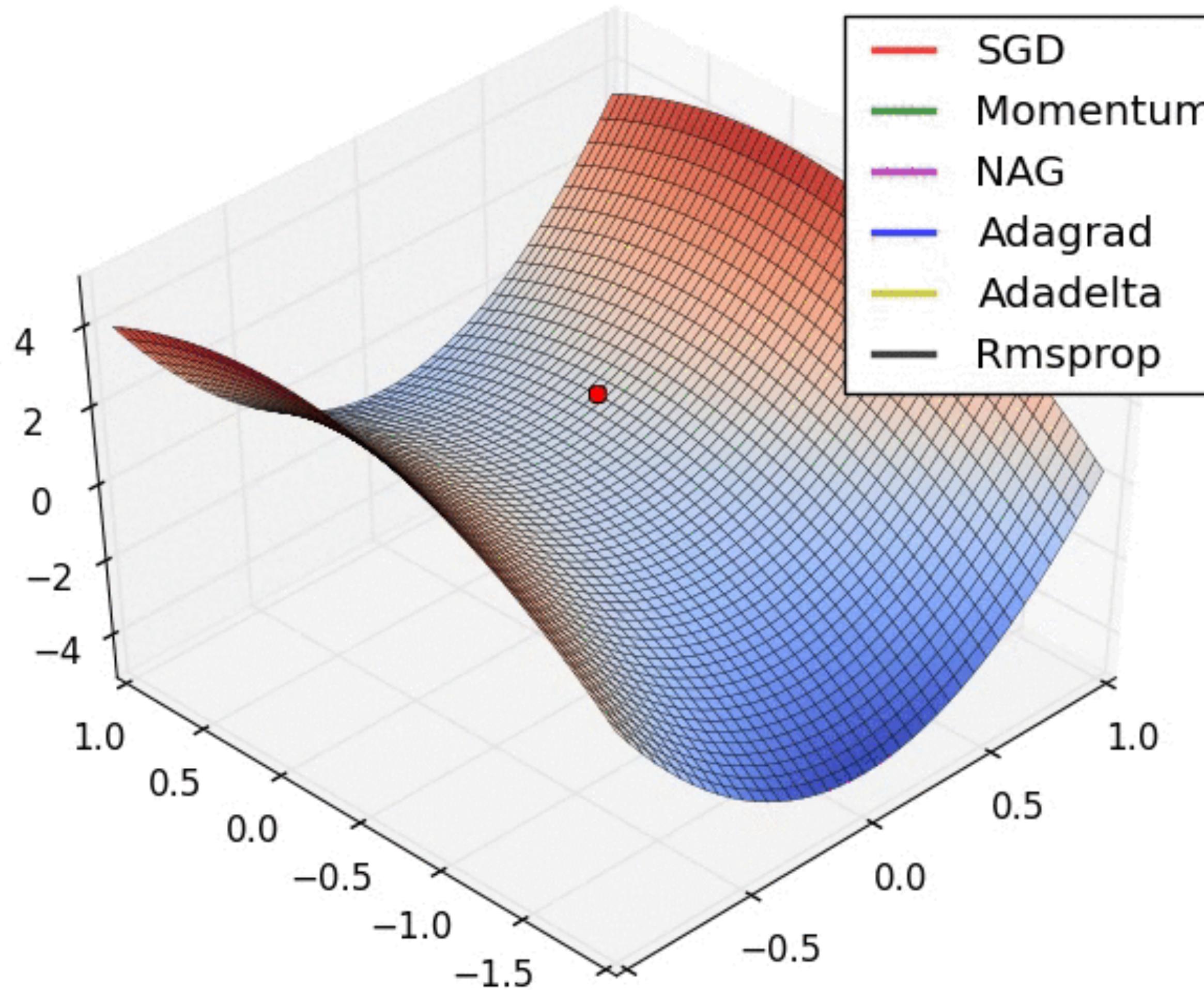
**ELU**

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

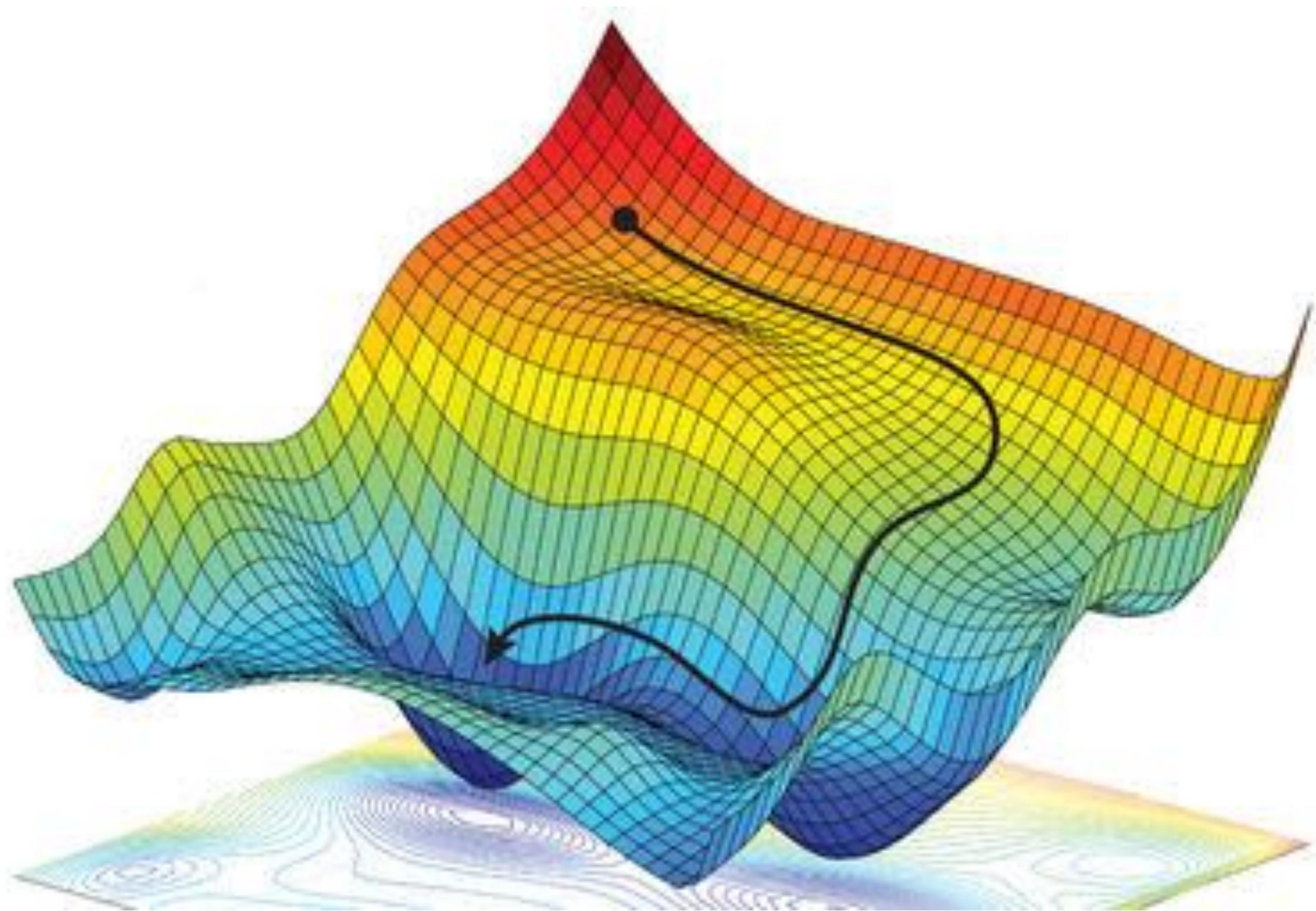


**But one can design their own activation functions!**

# Optimizers



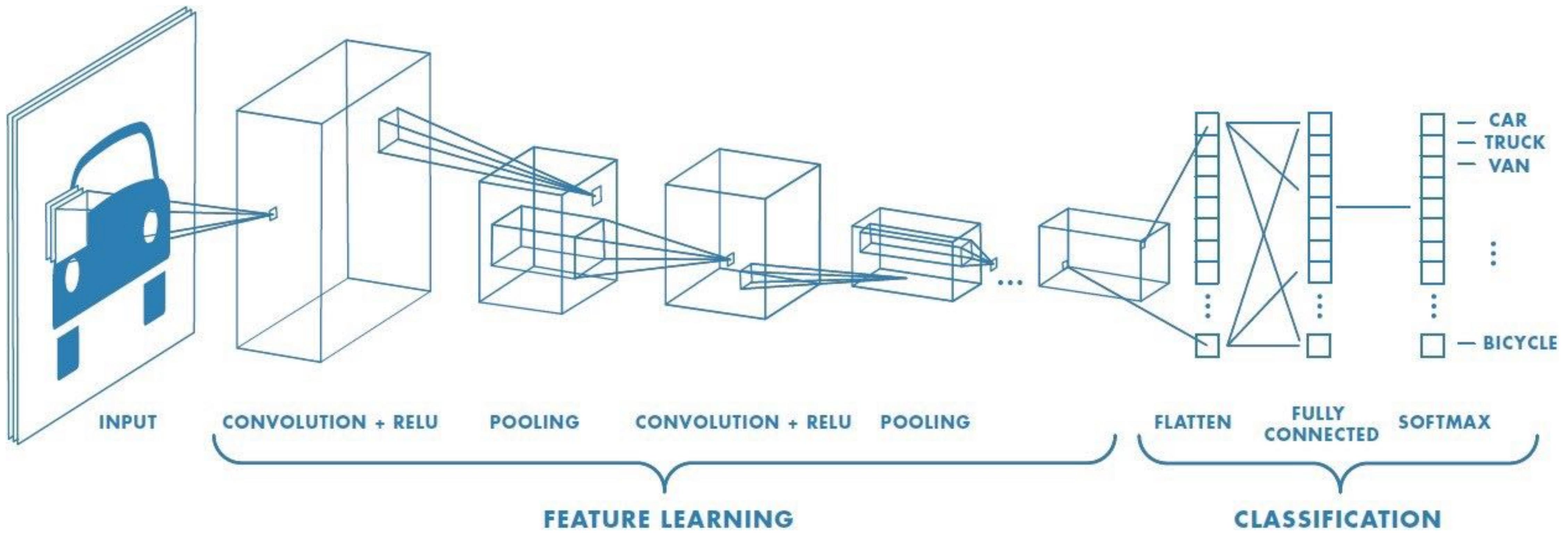
Source: [medium.com/analytics-vidhya](https://medium.com/analytics-vidhya)



# **Building blocks of DNN**

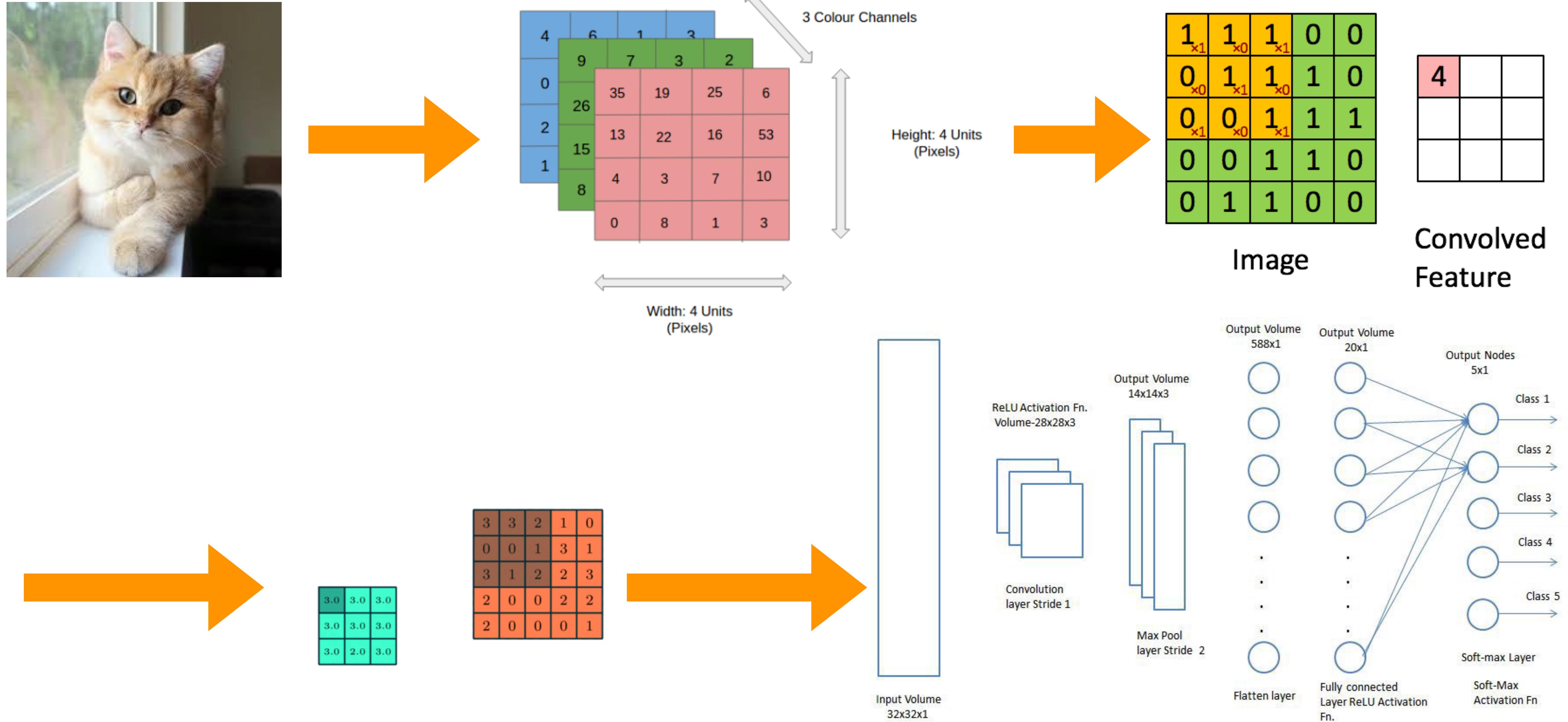
*CNN, Dense, LSTM, etc.*

# Convolution Neural Network (CNN)



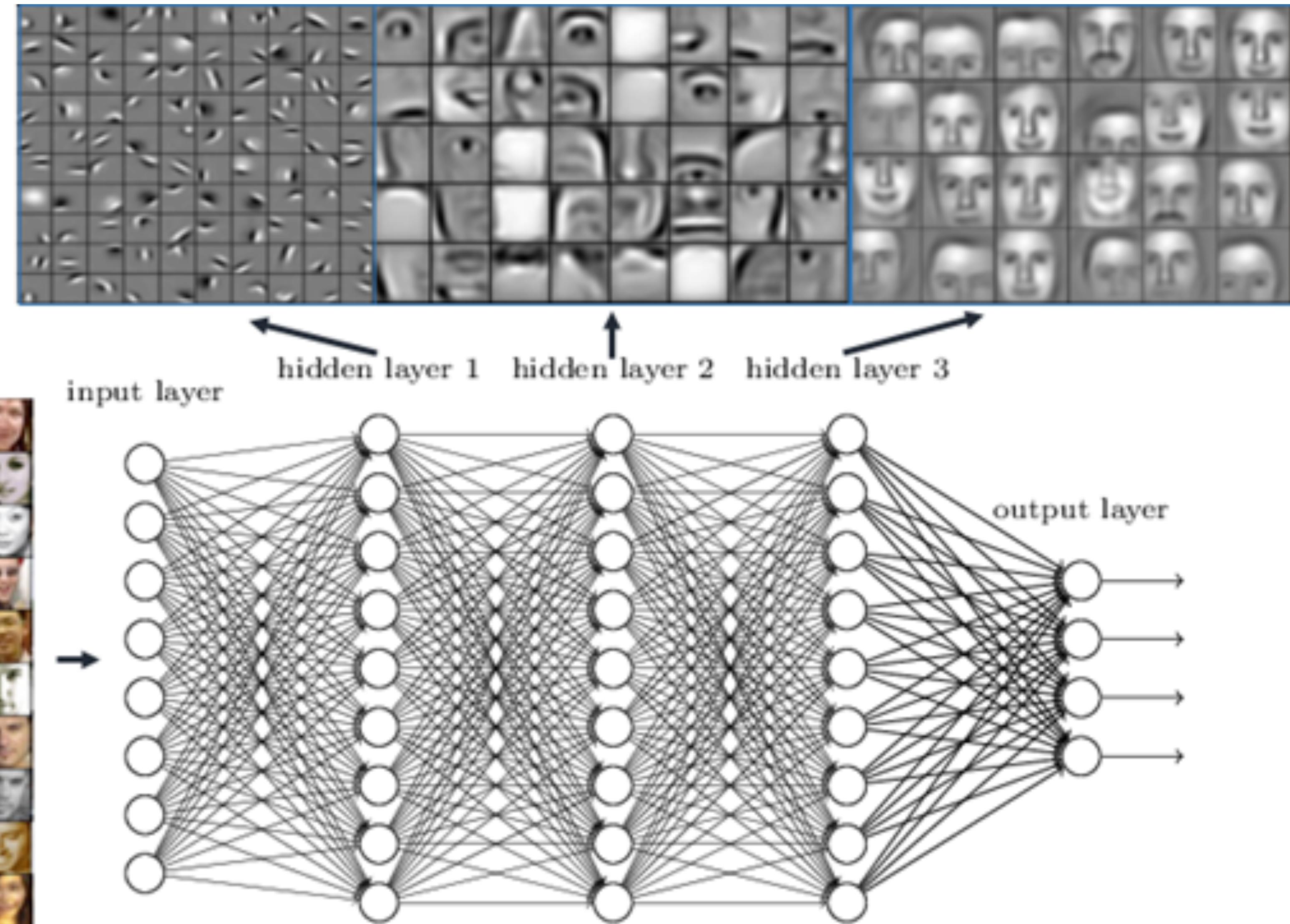
CNN: Inspired by the organization of visual cortex, which responds to stimuli only in a restricted region

# Convolution Neural Network (CNN)

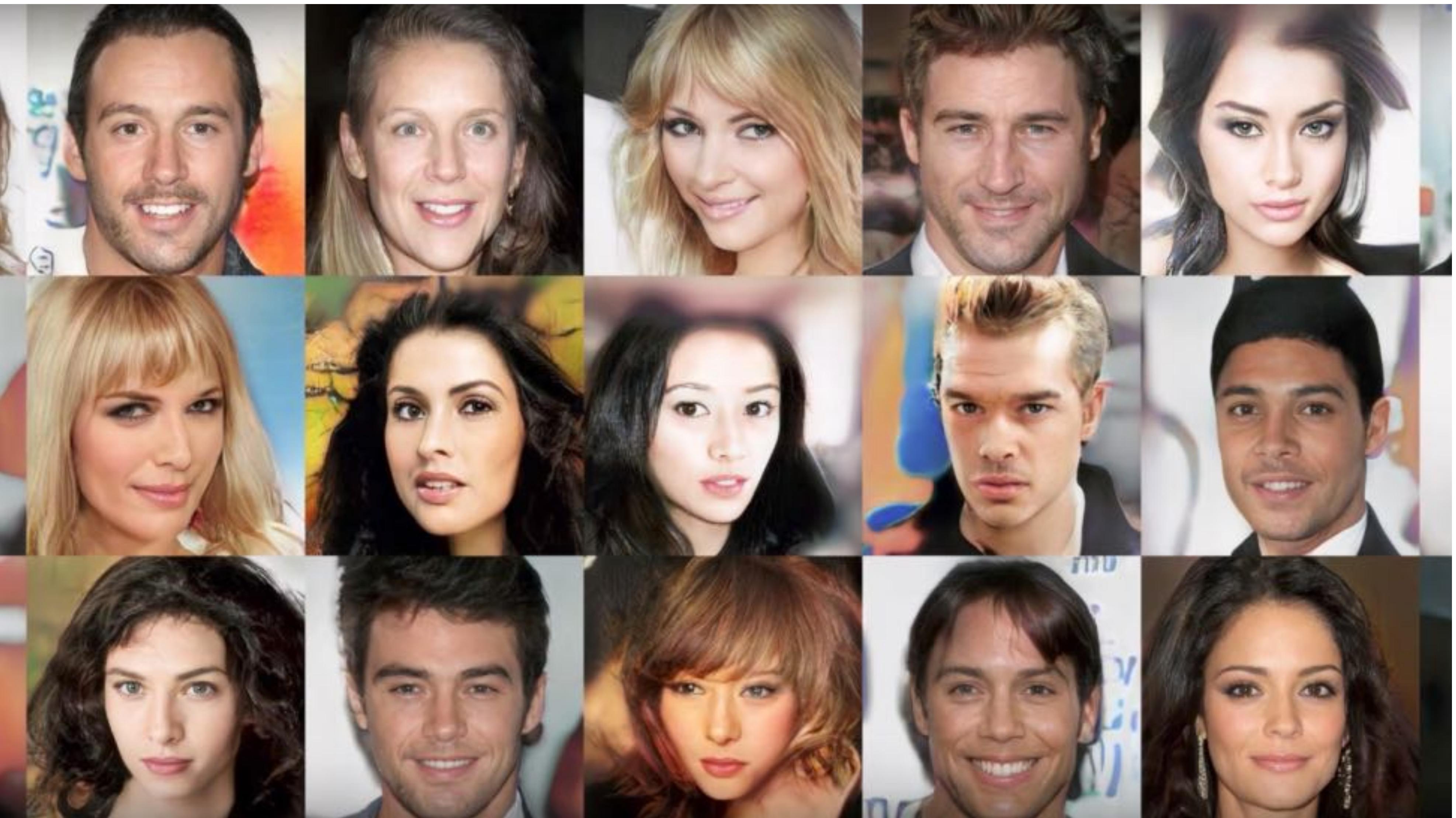


# CNNs are hierarchical

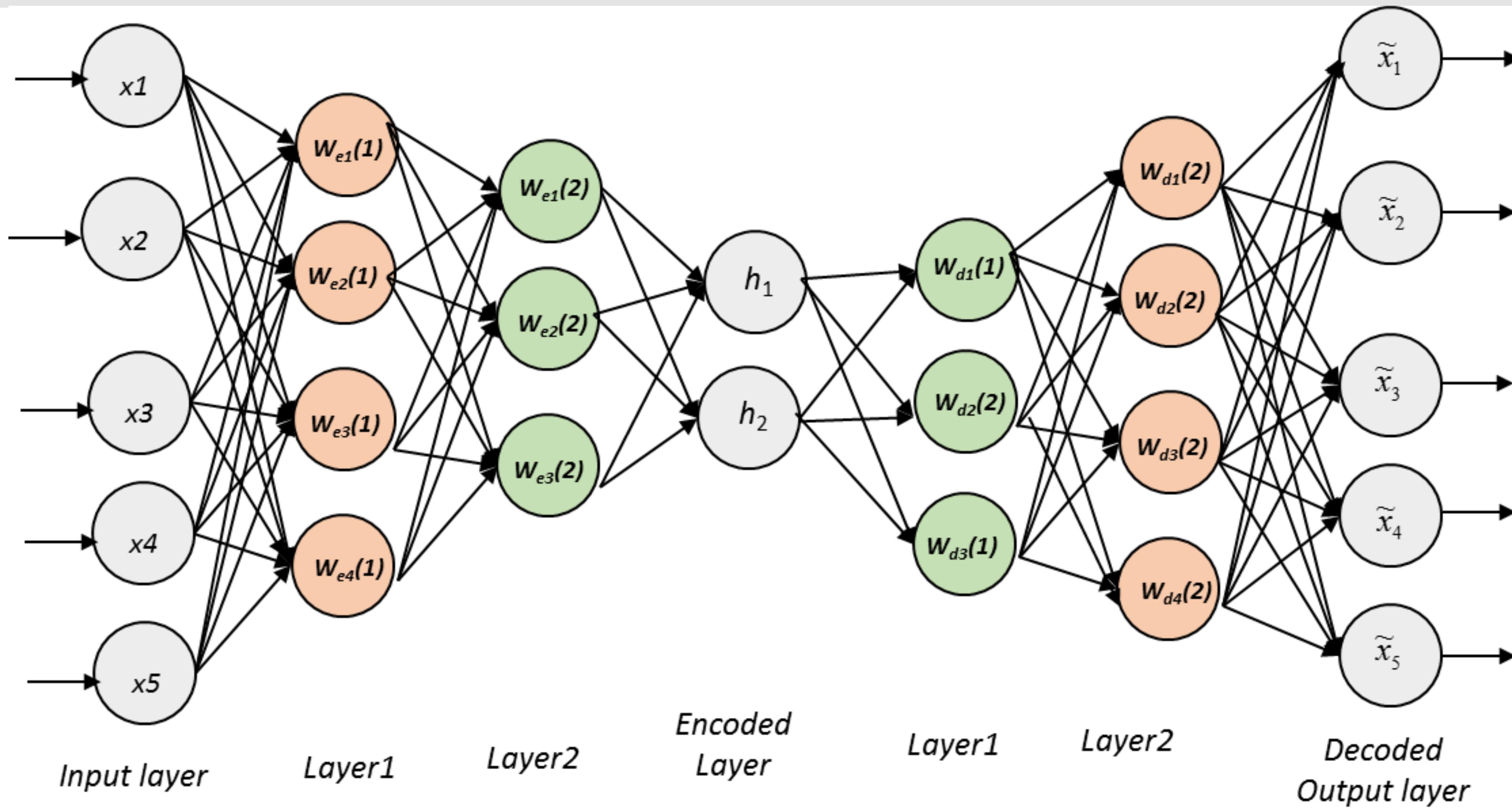
Deep neural networks learn hierarchical feature representations



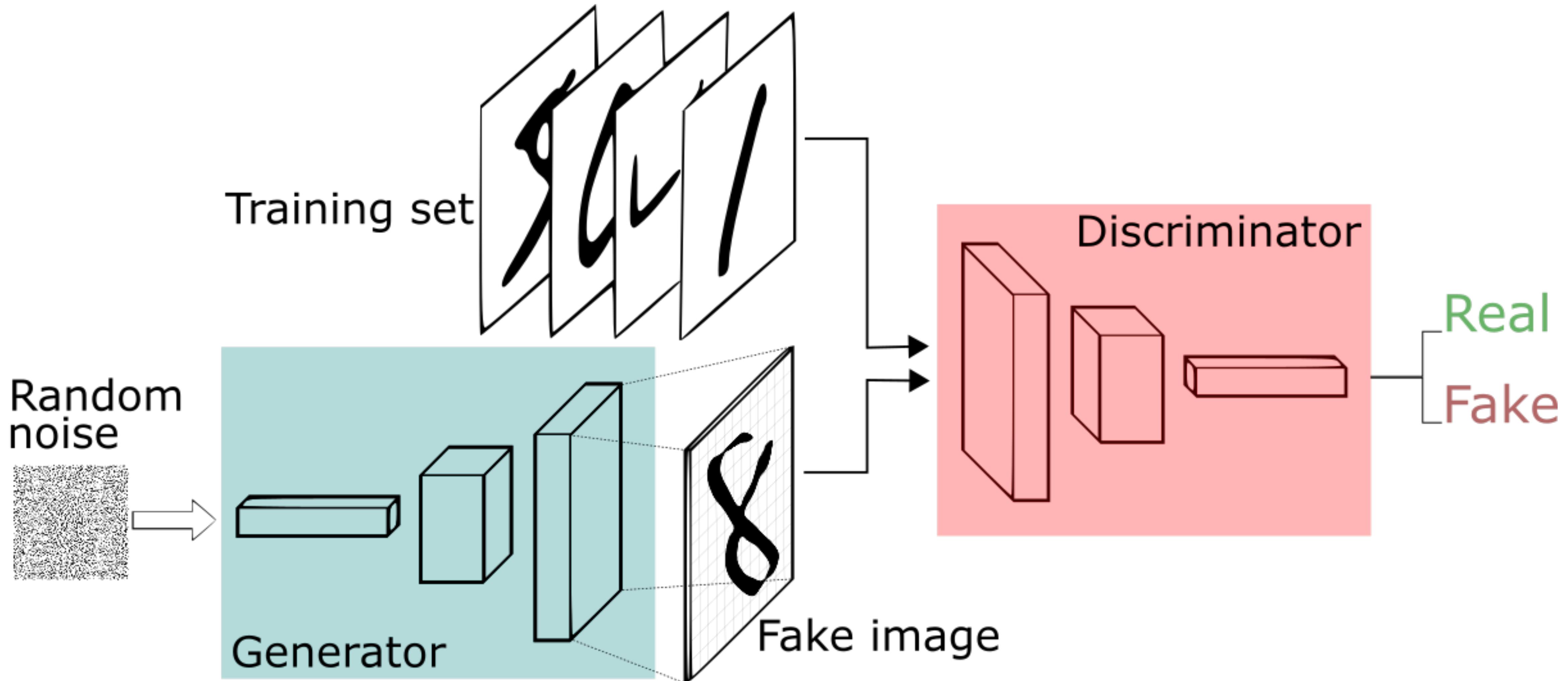
**How about generating images from labels?**



# Autoencoders



# Generative Adversarial Networks

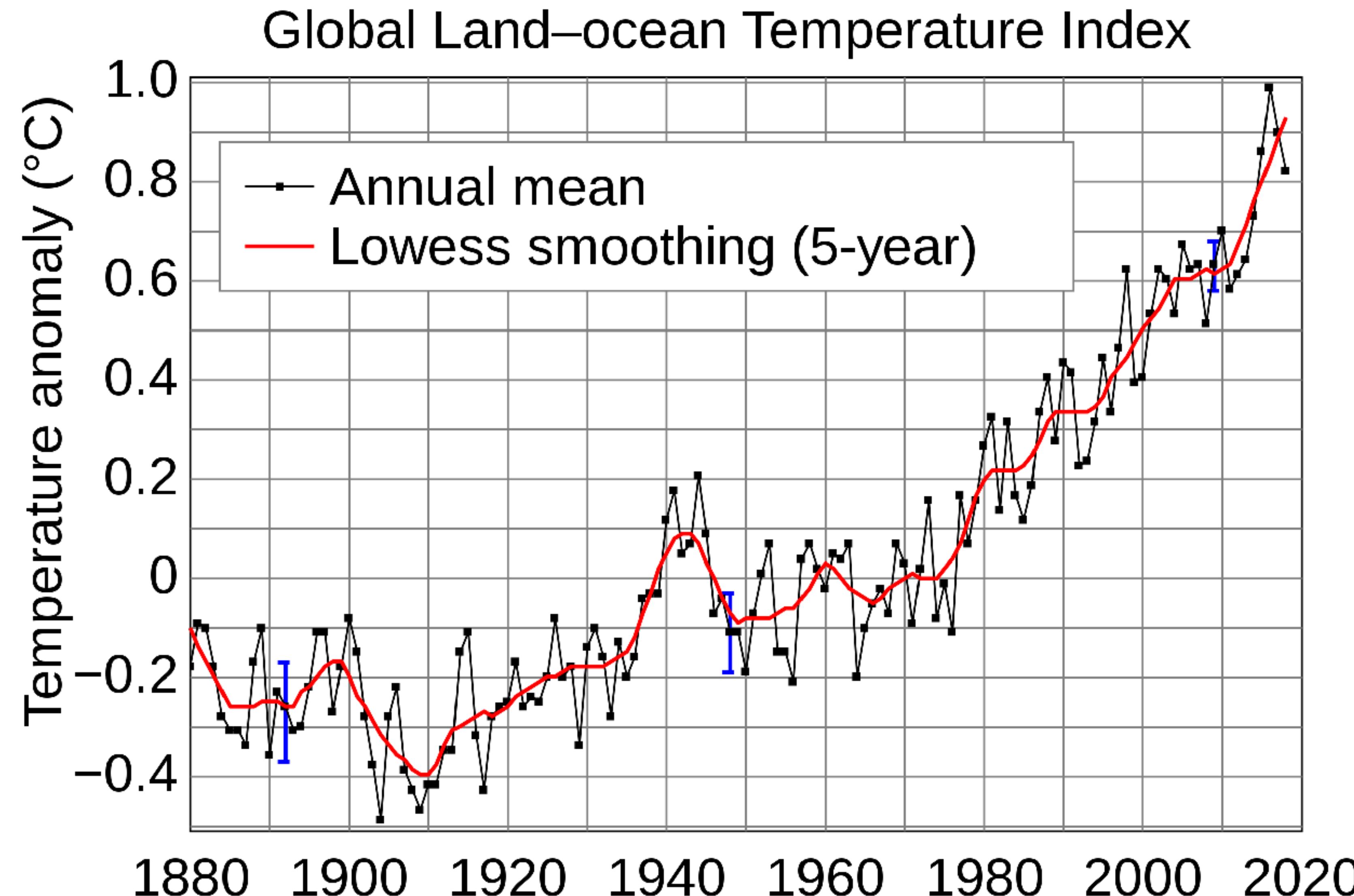




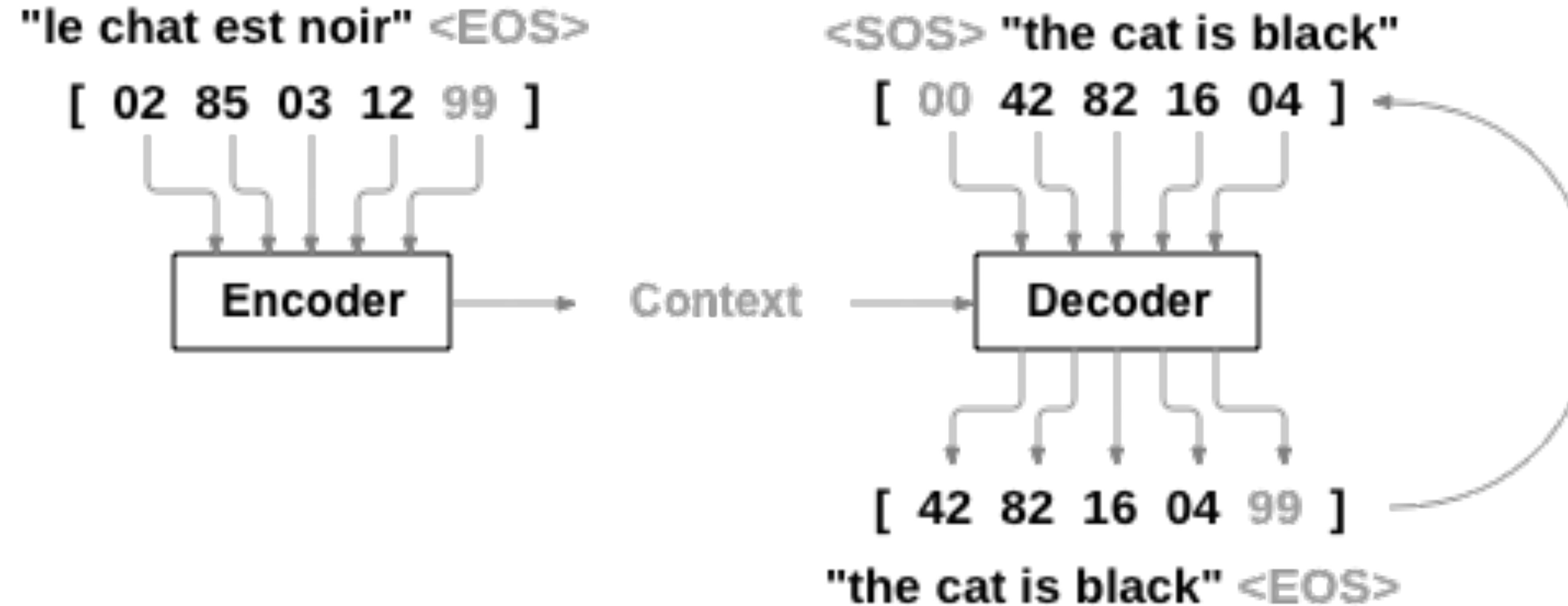
4x4

# Sequence Modeling

Image credit: NASA/Goddard



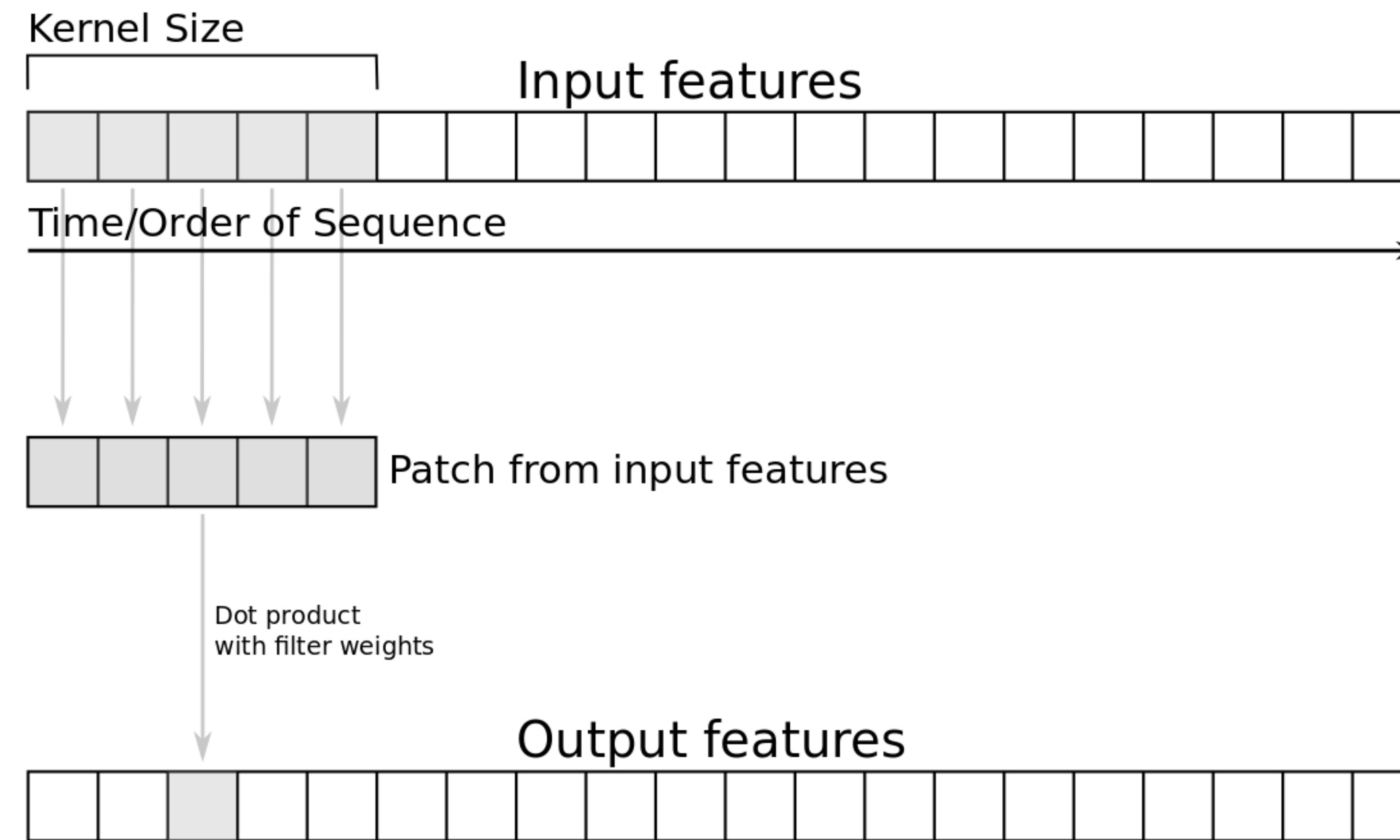
# Sequence Modeling: Recurrent Neural Networks

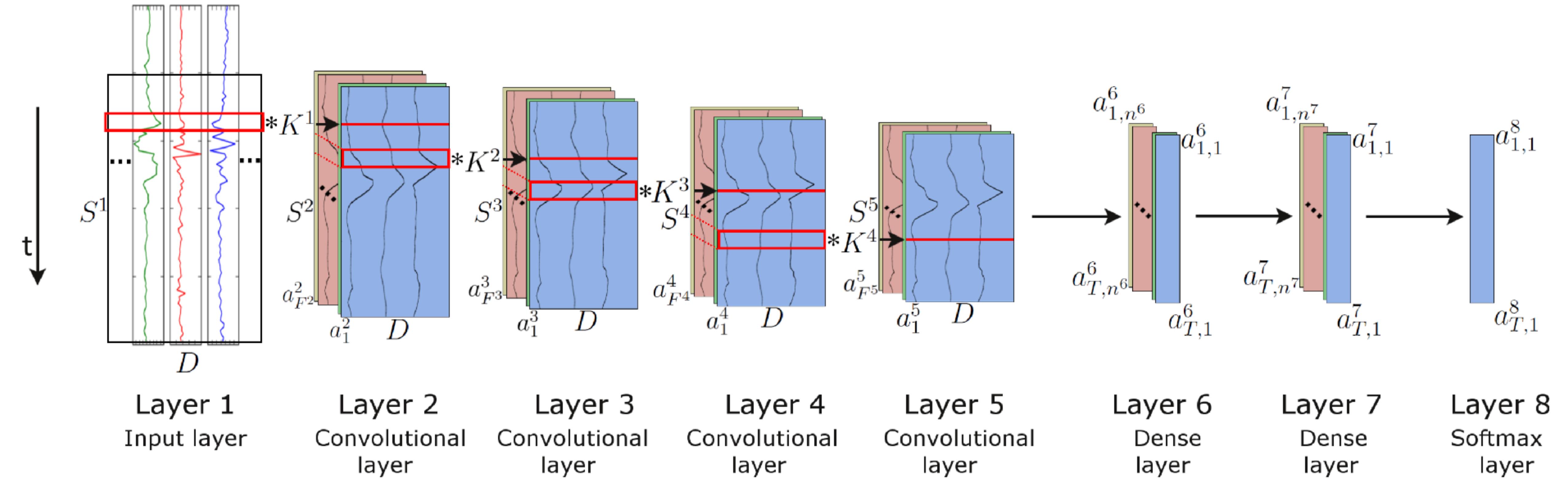


CNNs: spatial relevance

RNNs: temporal relevance

# 1D Convolution



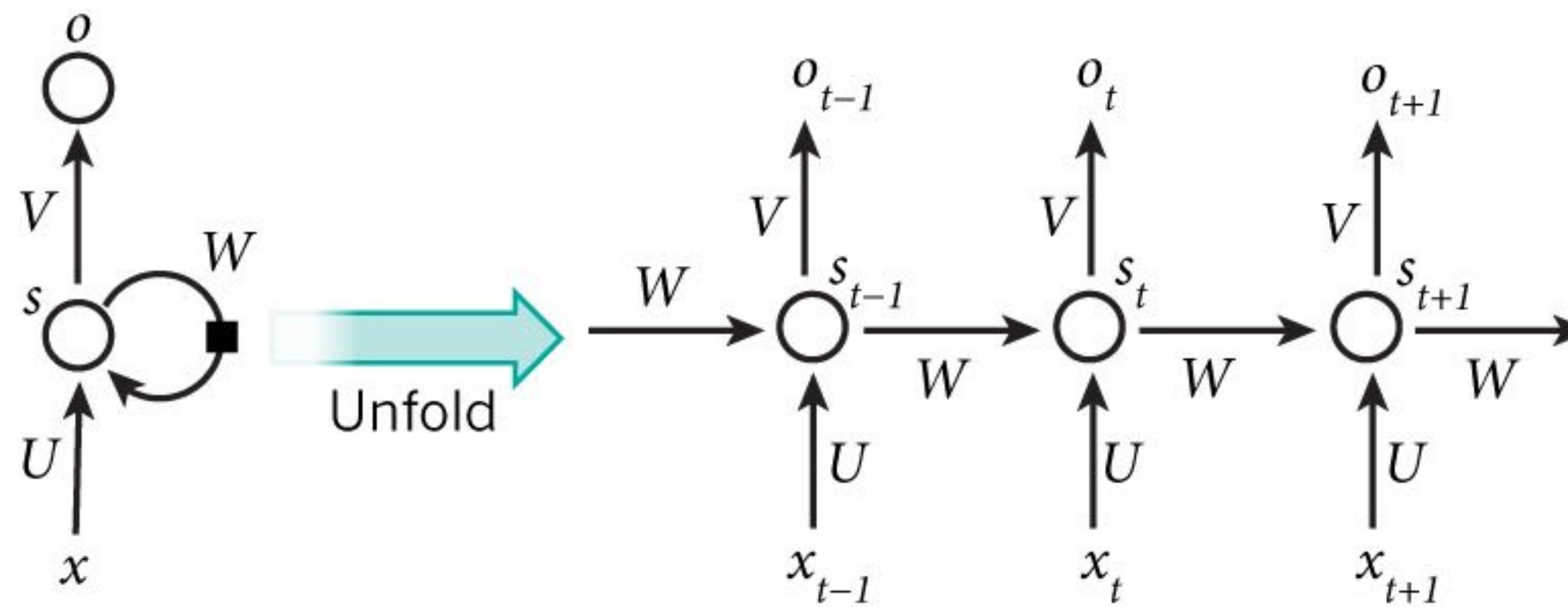


# Unfolded Computational Graph of RNNs

**Sequence**  $(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \dots, x^{(2)}, x^{(1)})$

**Recursive unfold**  $s^{(t)} = f(s^{(t-1)}; \theta)$

**Output Sequence**  $(o^{(t)}, o^{(t-1)}, o^{(t-2)}, \dots, o^{(2)}, o^{(1)})$

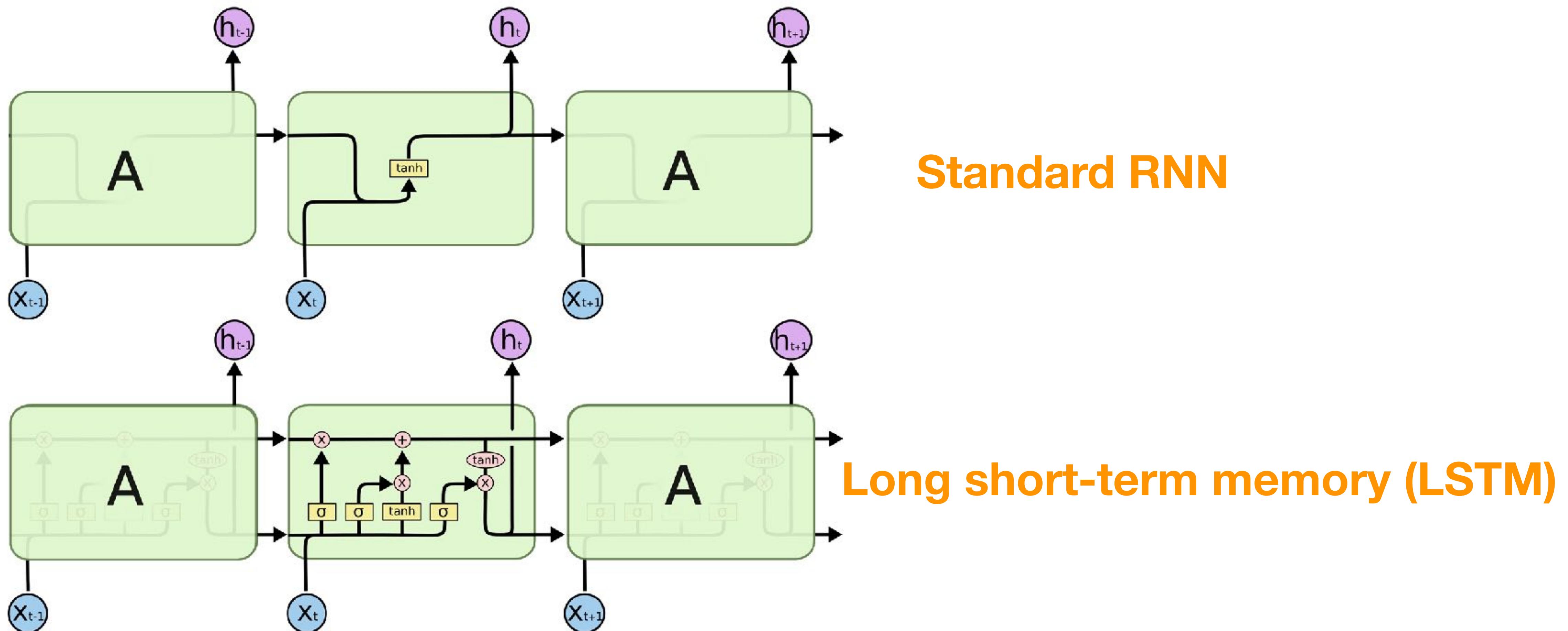


**CNNs: Backprop through layers**  
**RNNs: Backprop through time**

# The Problem of Long-term Dependencies

The clouds are in the *sky*.

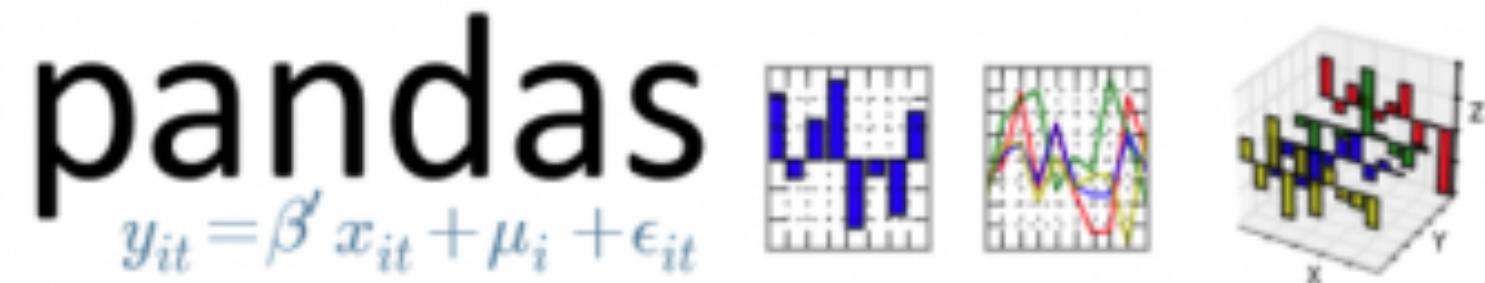
I grew up in France... I speak fluent *French*.



- ... ML/DL are built upon rigorous math theories
- ... ML/DL requires high-performance computing
- ... But we could just stand upon the shoulders of the giants



**IP[y]:** IPython  
Interactive Computing





theano

