Deep Learning

Mahdi Abolfazli Esfahani

Nanyang Technological University

Human-Centered AI

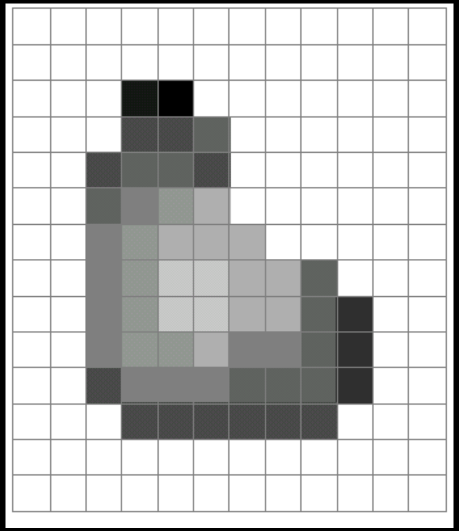
$$\begin{tikzcd} 90\% & \text{Human Needed} \arrow[l, "\text{No}"'] \arrow[r, "\text{Yes}"] & 10\% \end{tikzcd}$$

Solve the perception-control problem where **possible**:

And where **not possible**: involve the human

WHAT IS AN IMAGE?

- A grid (matrix) of intensity values

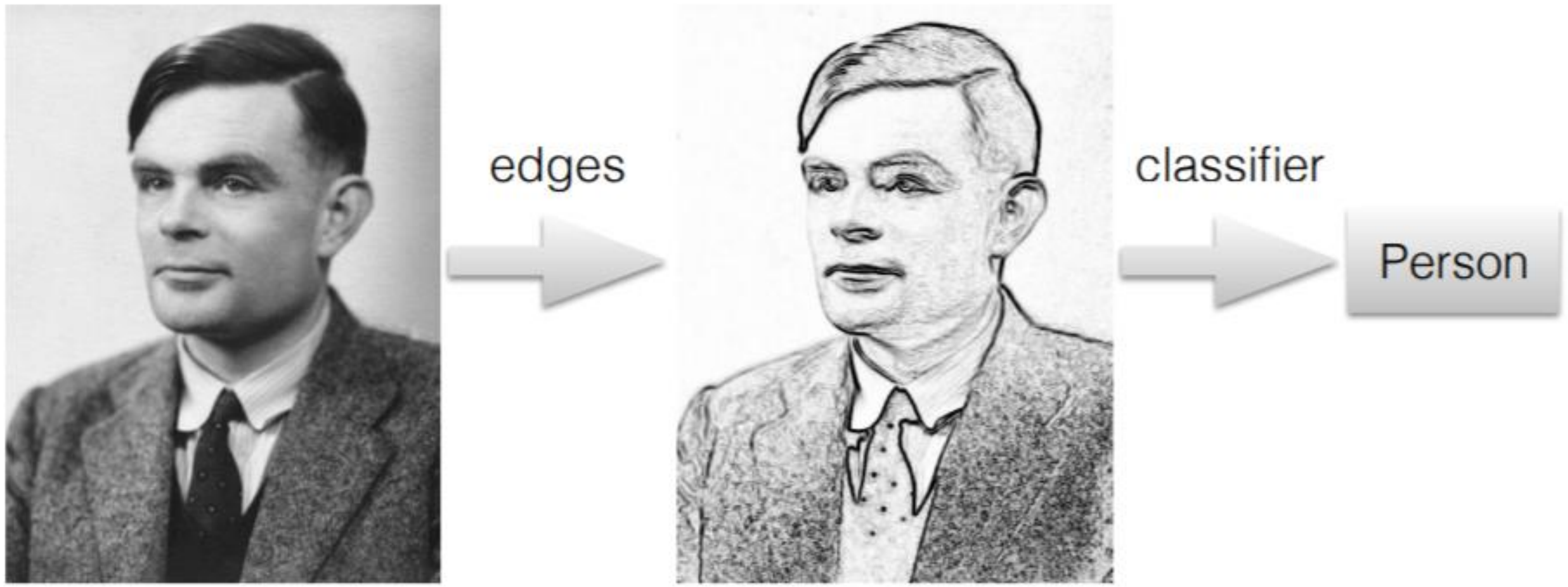


=

255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

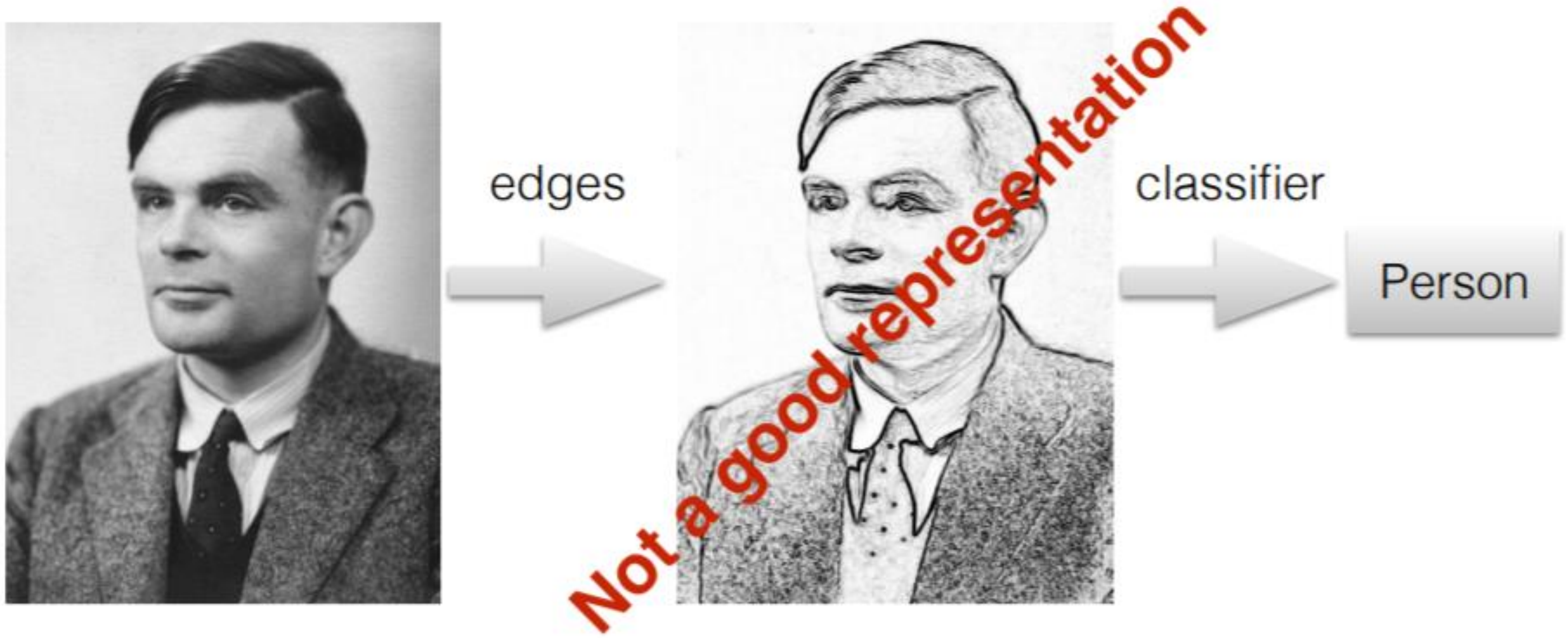
common to use one byte per value: 0 = black, 255 = white
Gray Image

How to teach a machine ?



(or any other **hand-crafted** features)

How to teach a machine ?



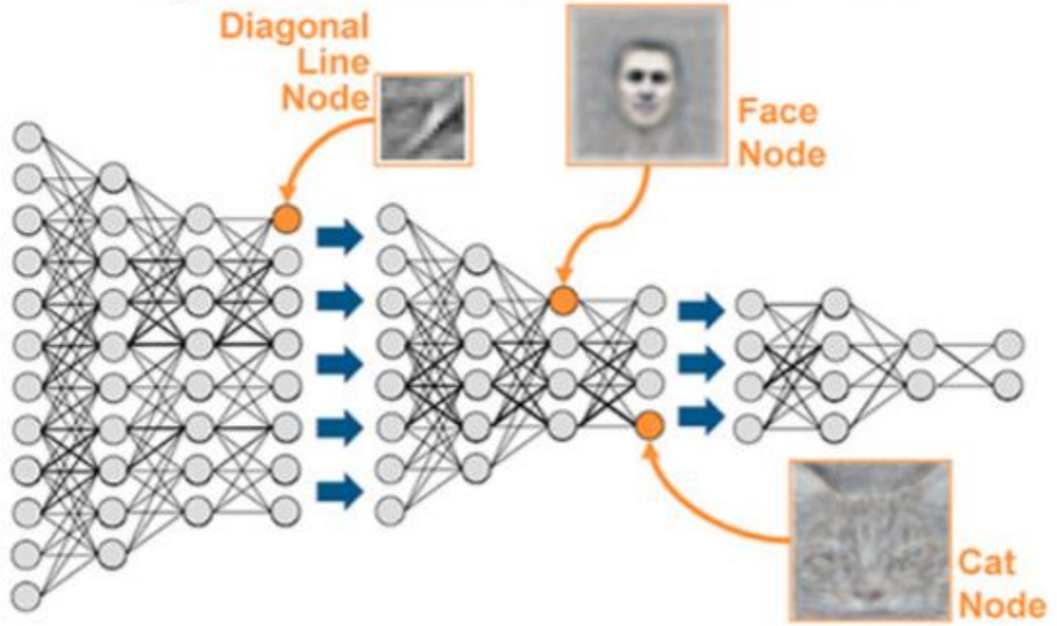
(or any other **hand-crafted** features)

What is deep learning ?

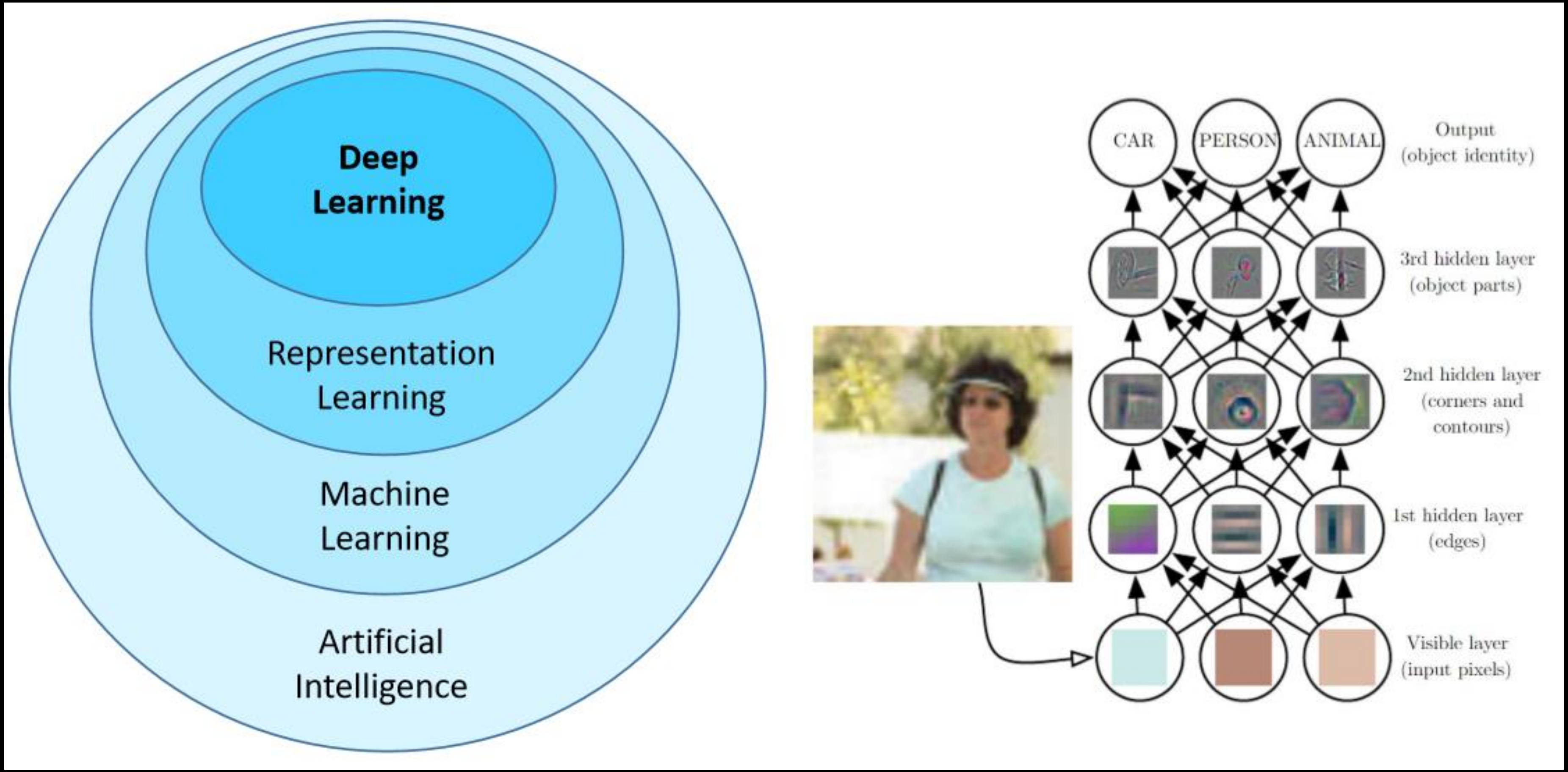
- Representation learning method
  Learning good features automatically from raw data

- Learning representations of data with multiple levels of abstraction

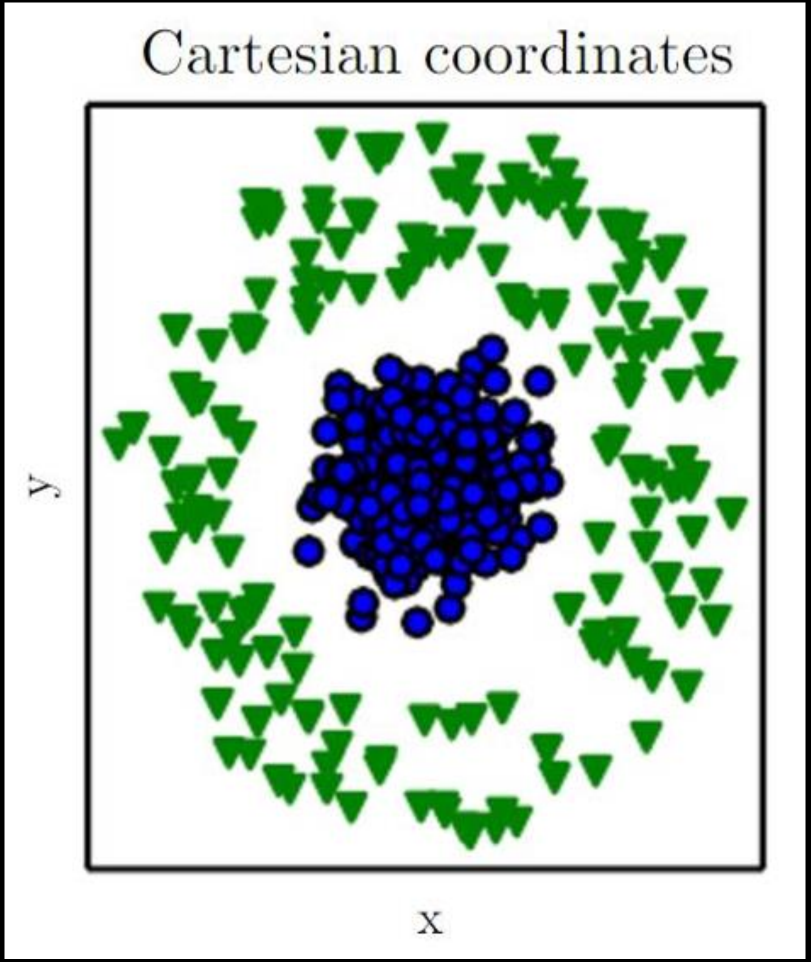
Google's cat detection neural network



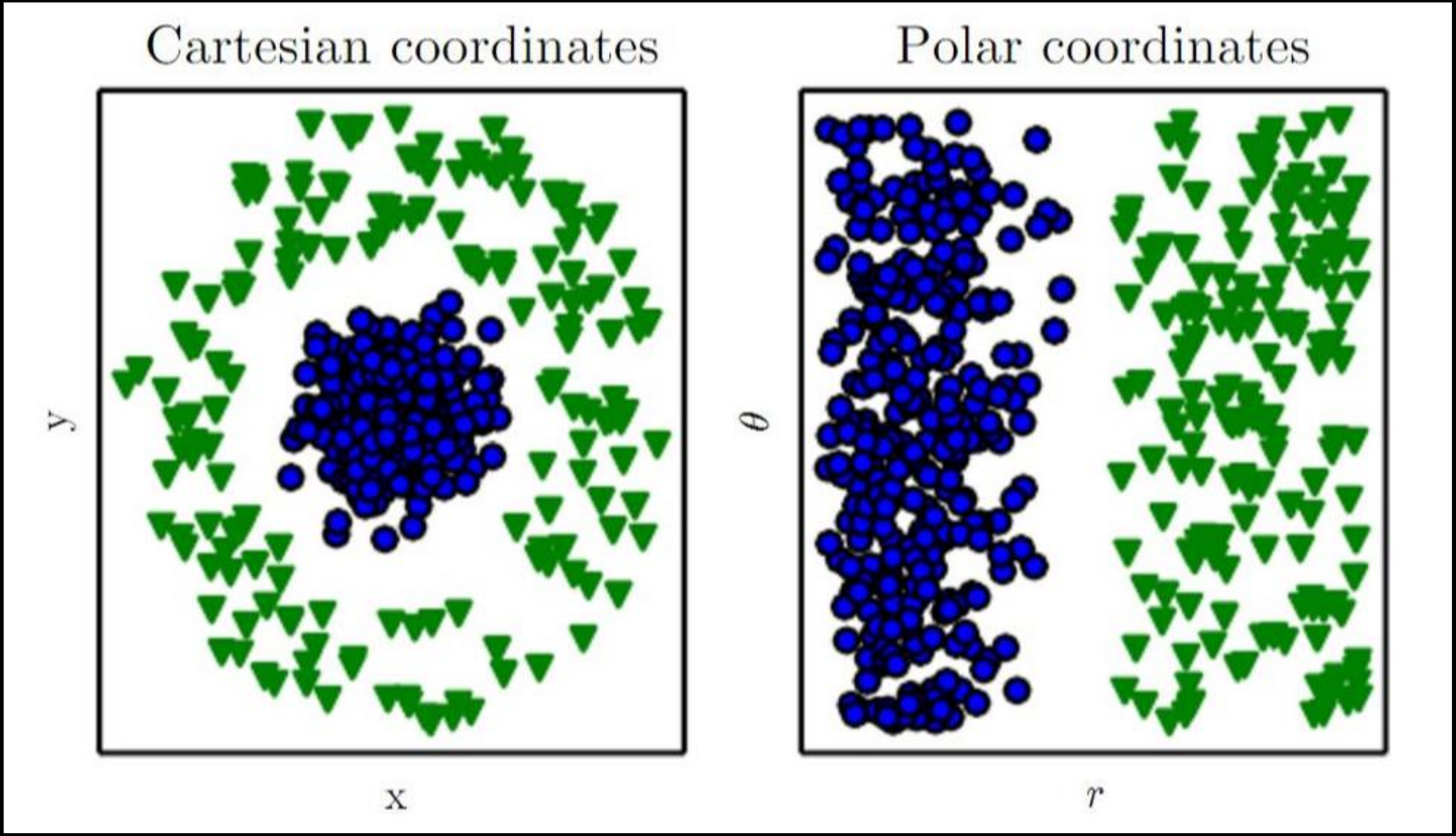
Deep Learning Is Representation Learning



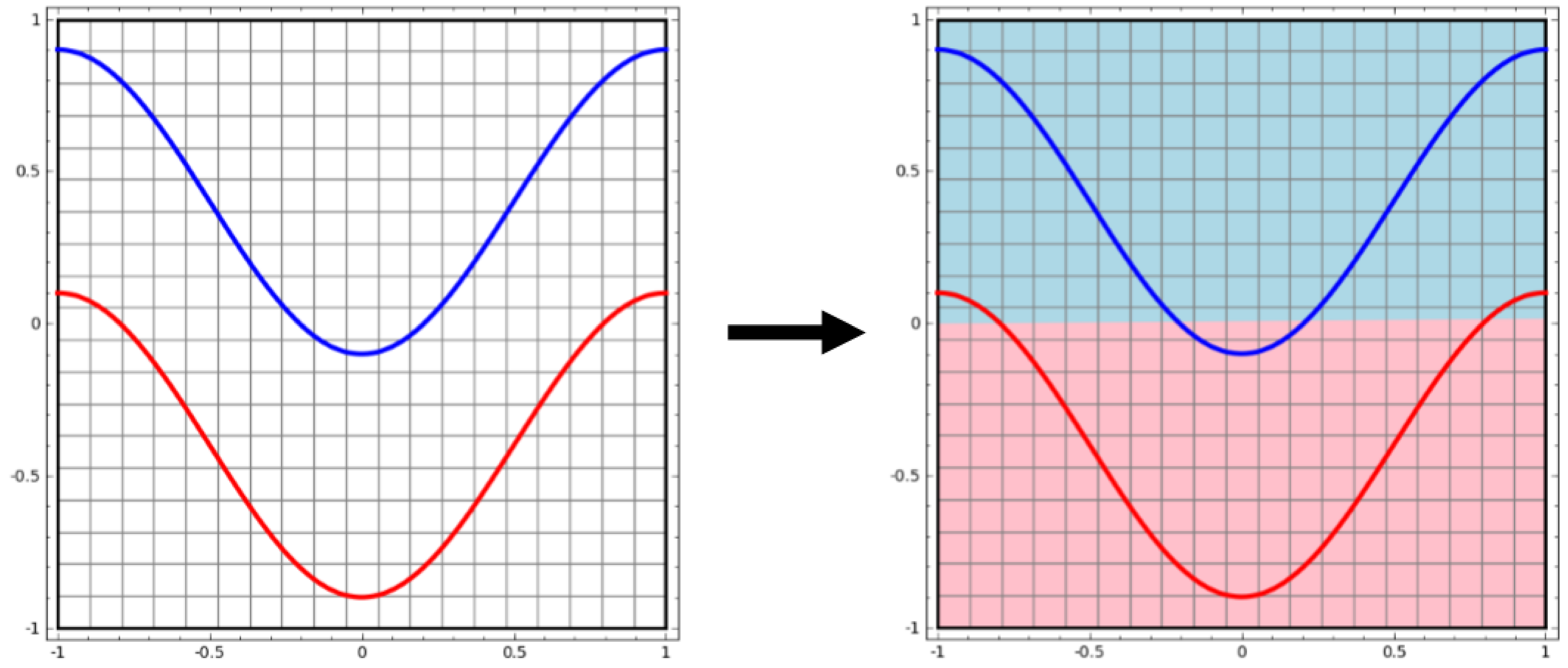
Why Representation Is Important?



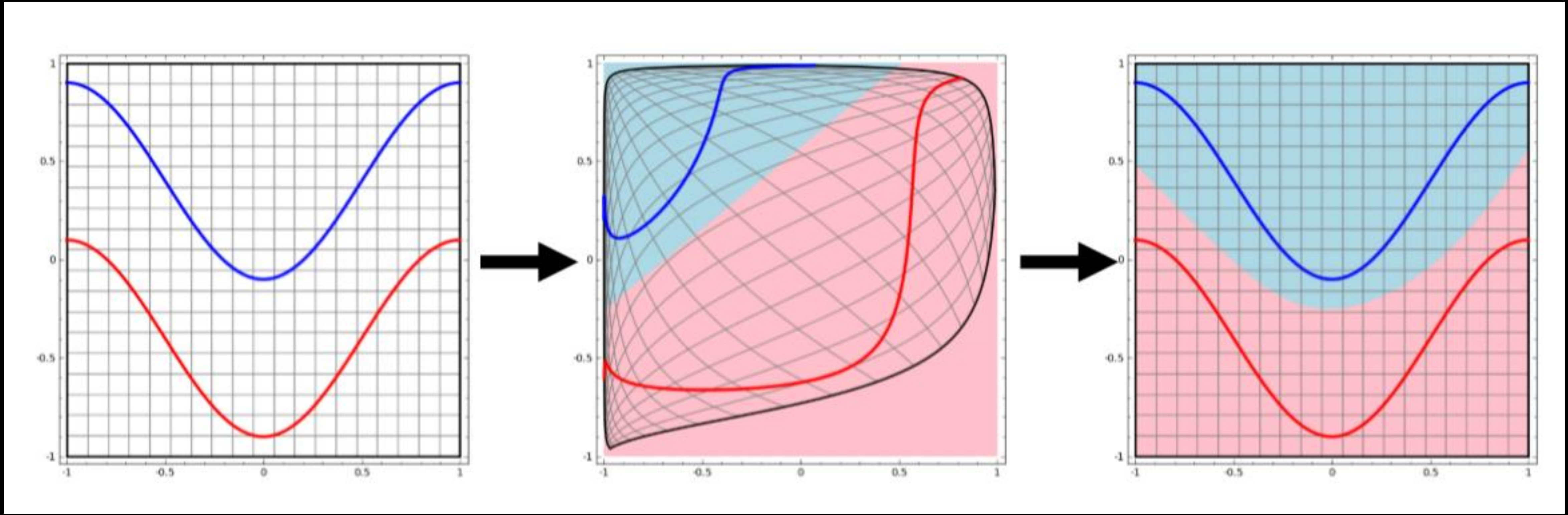
Why Representation Is Important?



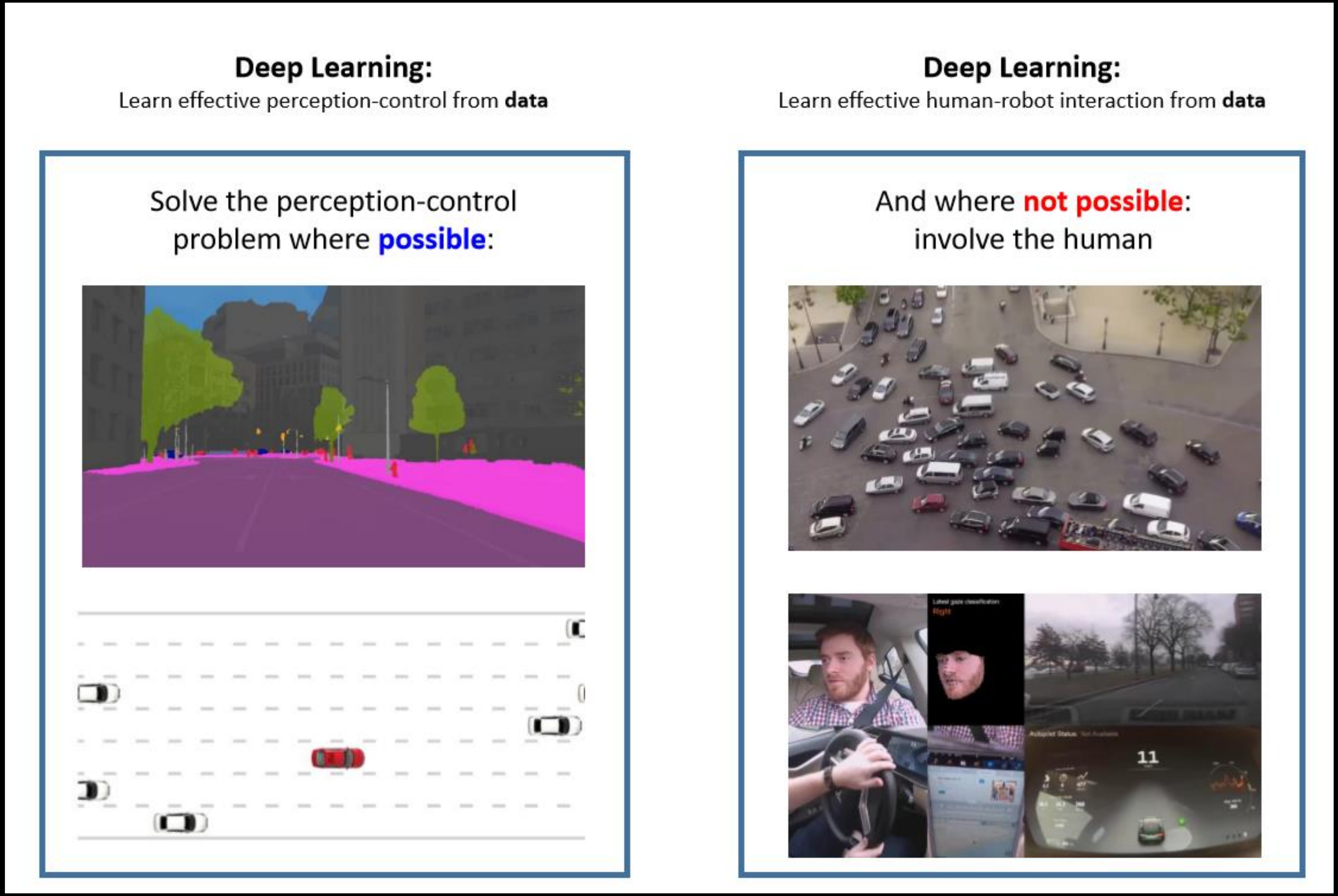
Why Representation Is Important?



Why Representation Is Important?



Why Deep Learning?



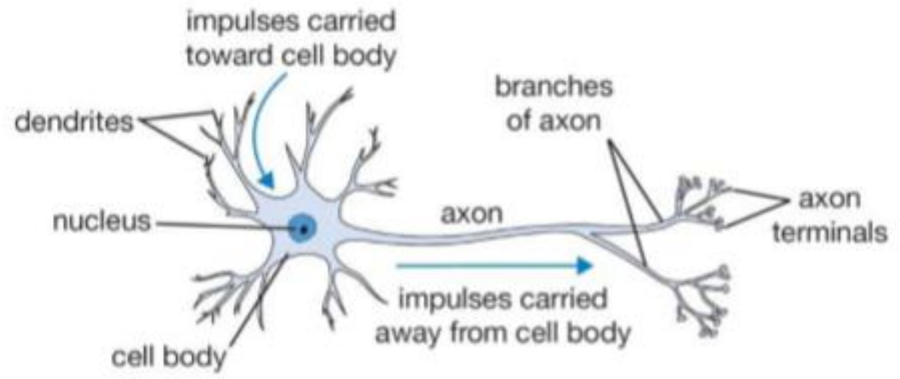
Deep Learning:
Learn effective perception-control from **data**

Solve the perception-control problem where **possible**:

Deep Learning:
Learn effective human-robot interaction from **data**

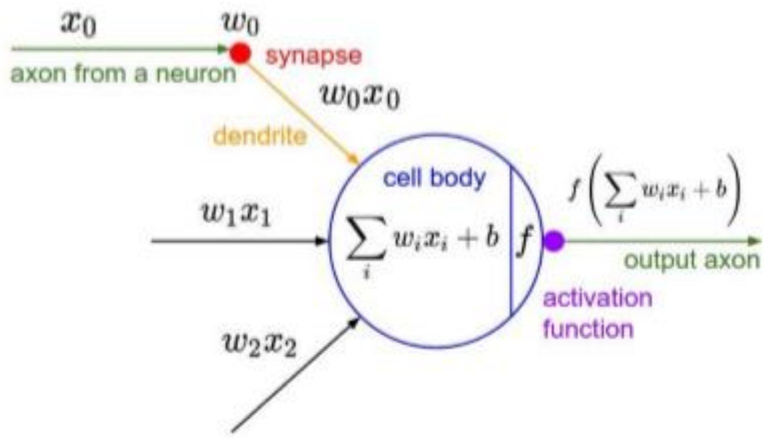
And where **not possible**: involve the human

Biological Inspiration For Computation



impulses carried toward cell body
branches of axon
dendrites
nucleus
axon
axon terminals
impulses carried away from cell body
cell body

- **Neuron:** computational building block for the brain

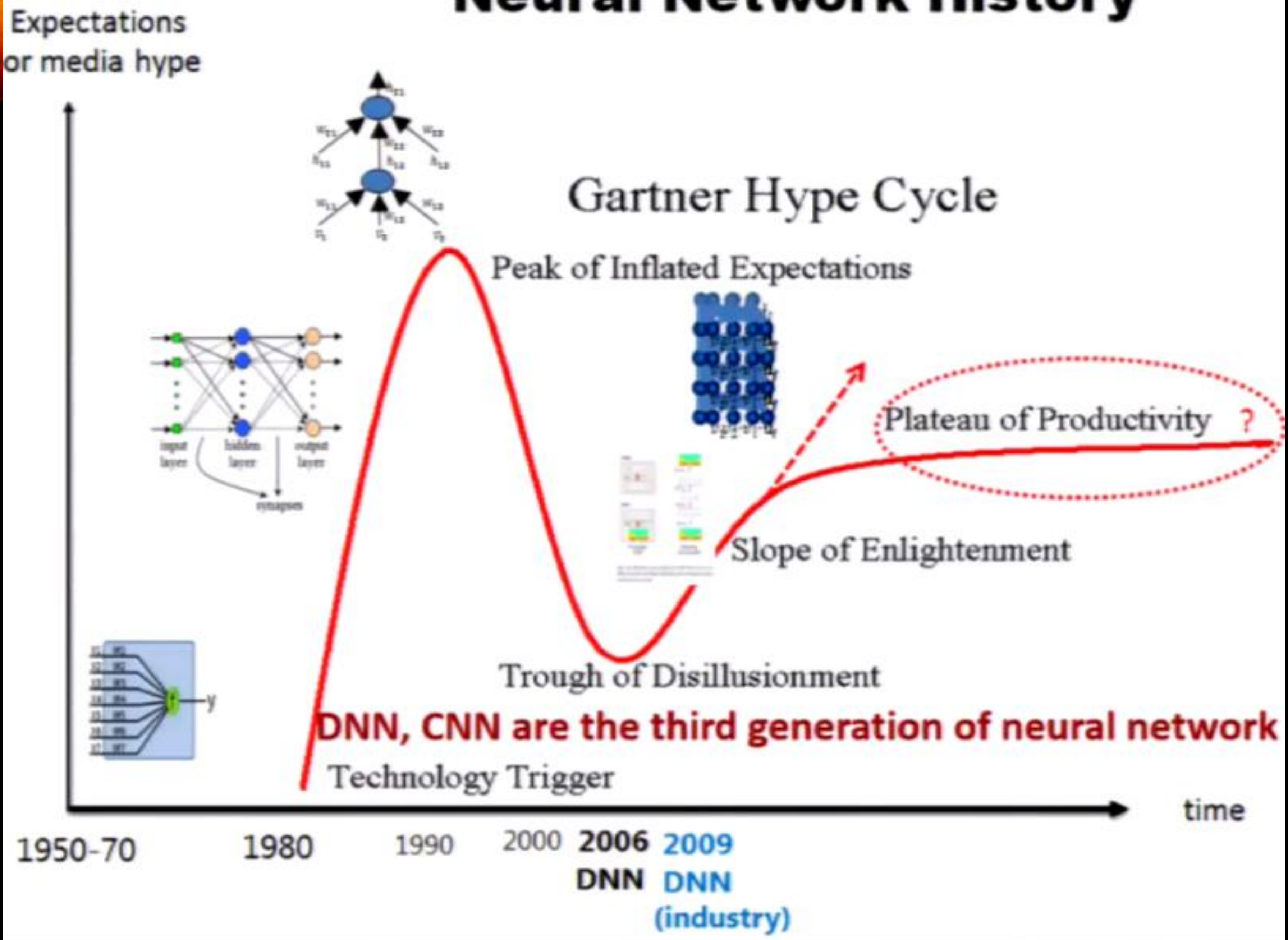


- **(Artificial) Neuron:** computational building block for the "neural network"

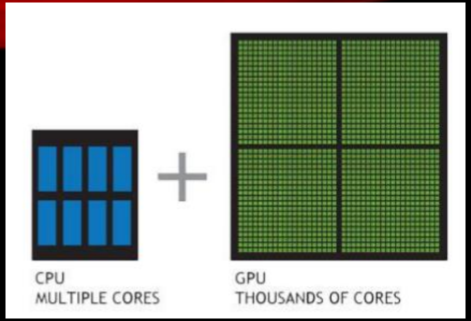
Biological Inspiration For Computation

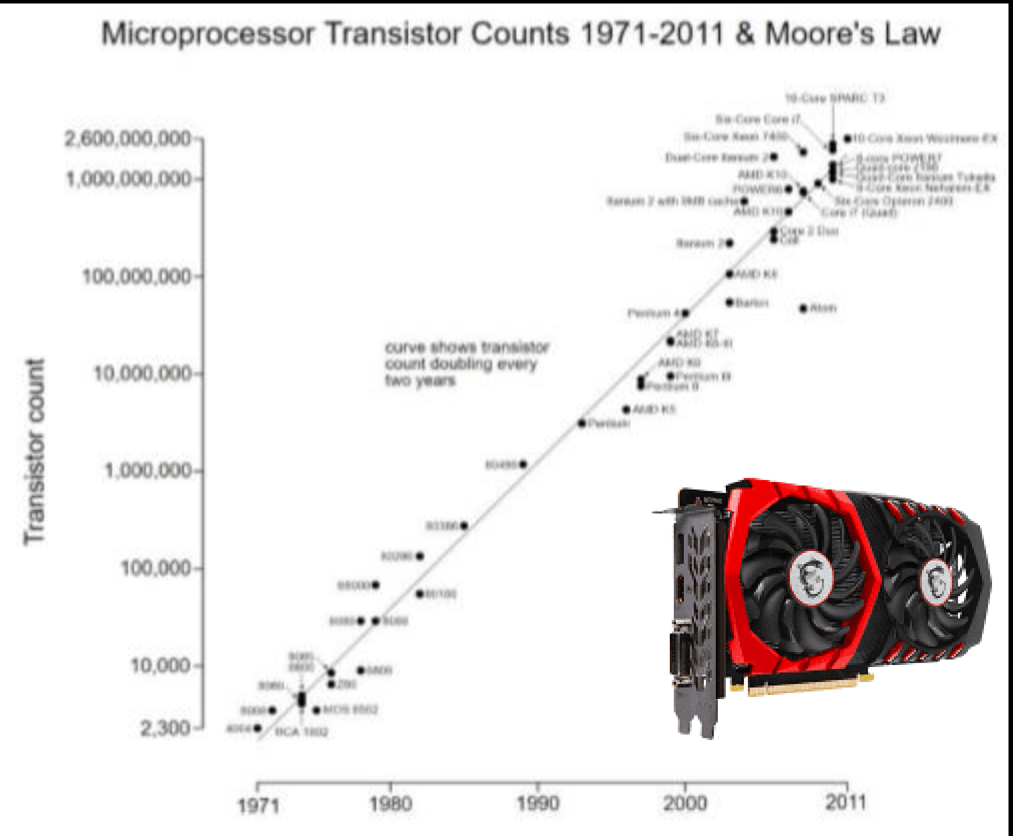
- **Parameters:** Human brains have ~10,000,000 times synapses than artificial neural networks.
- **Topology:** Human brains have no “layers”. Topology is complicated.
- **Async:** The human brain works asynchronously, ANNs work synchronously.
- **Learning algorithm:** ANNs use gradient descent for learning. Human brains use … (we don’t know)
- **Processing speed:** Single biological neurons are slow, while standard neurons in ANNs are fast.
- **Power consumption:** Biological neural networks use very little power compared to artificial networks
- **Stages:** Biological networks usually don't stop / start learning. ANNs have different fitting (train) and prediction (evaluate) phases

Neural Network History



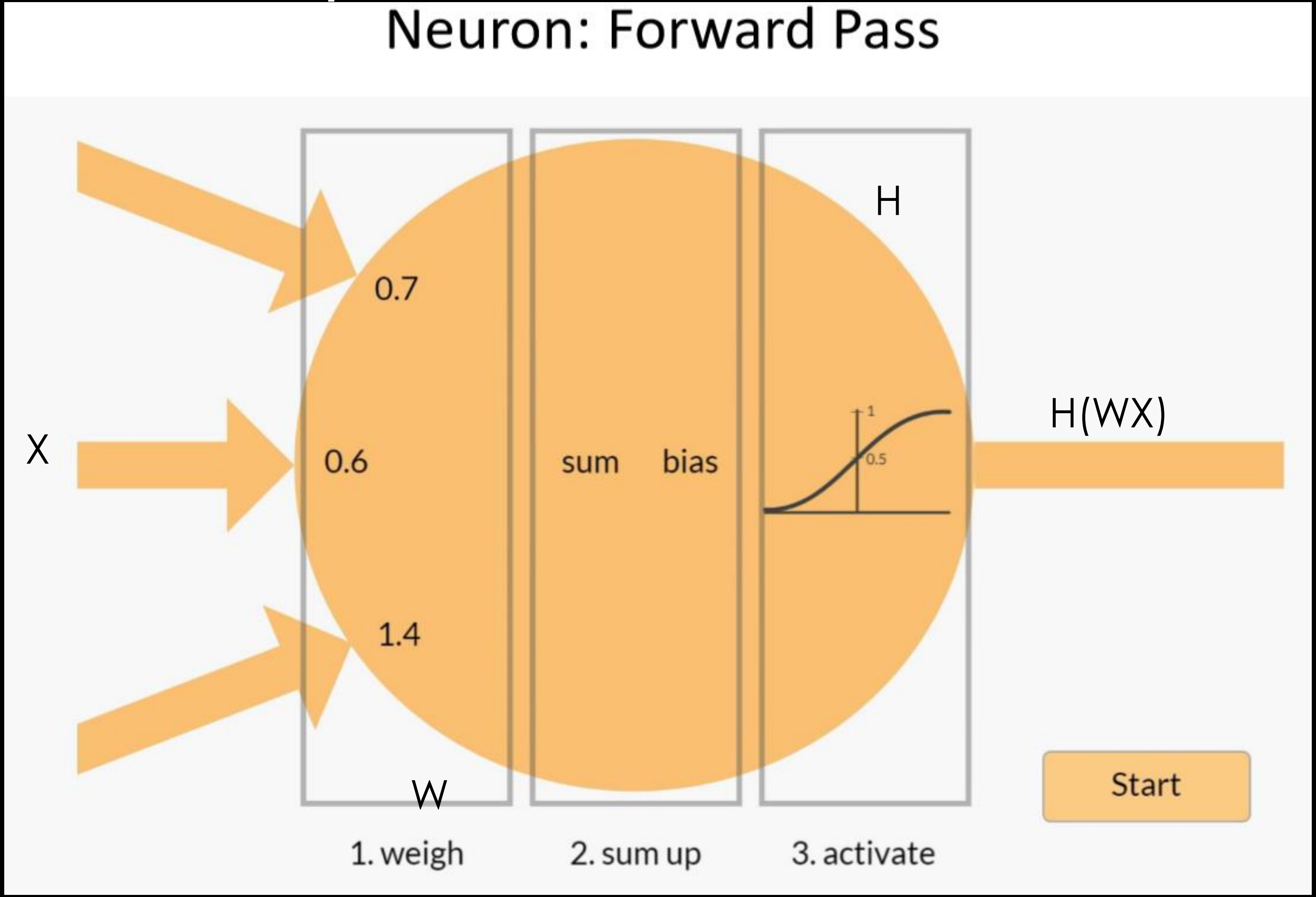
Deep Learning Breakthroughs: What Changed?



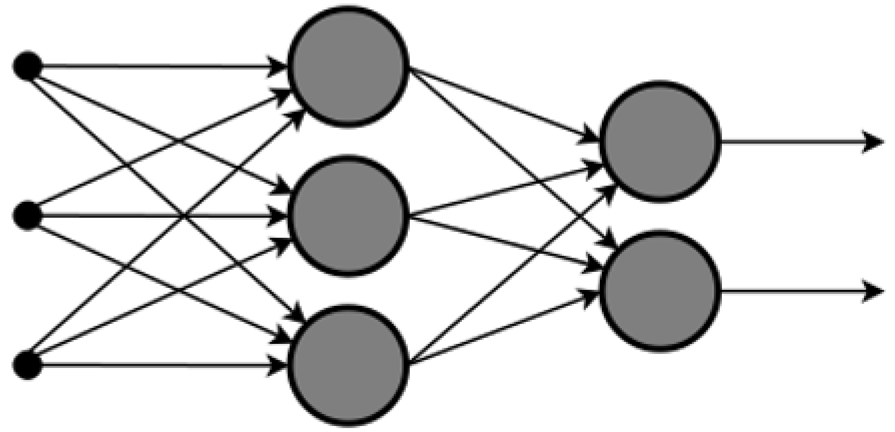


- **Compute** CPUs, GPUs
- **Organized large(-ish) datasets** Imagenet
- **Algorithms and research:** Backprop, CNN, LSTM
- **Software and Infrastructure** Git, ROS, PR2, AWS, Amazon Mechanical Turk, TensorFlow, …
- **Financial backing of large companies** Google, Facebook, Amazon, …

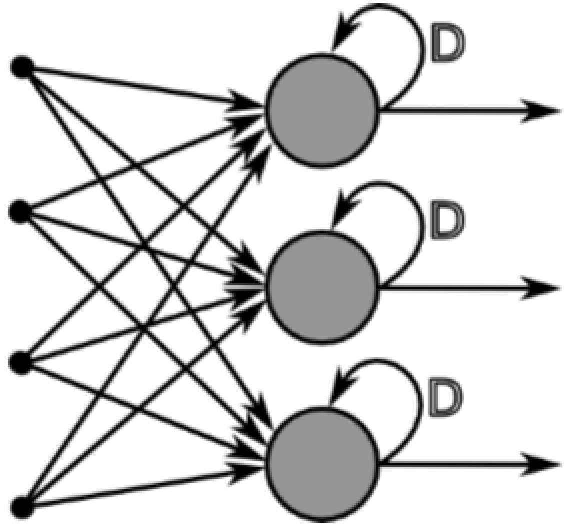
Biological Inspiration For Computation



Combining Neurons Into Layers



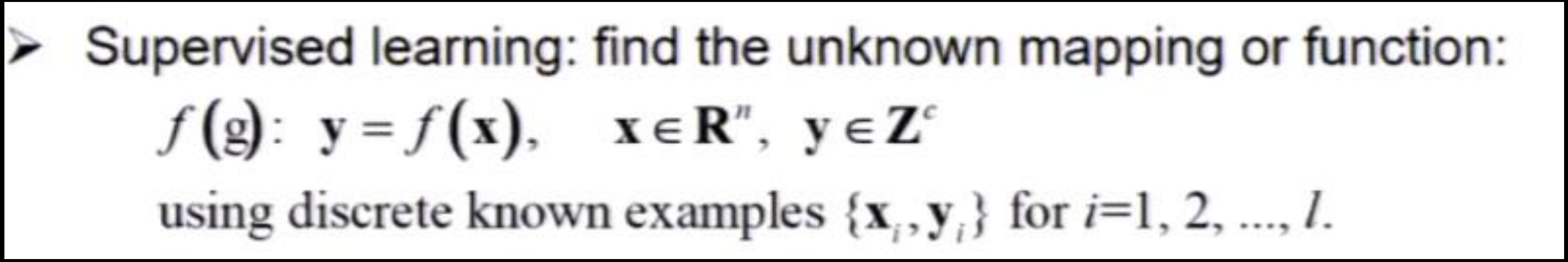
Feed Forward Neural Network



Recurrent Neural Network

- Have state memory
- Are hard to train

Understanding Deep Learning

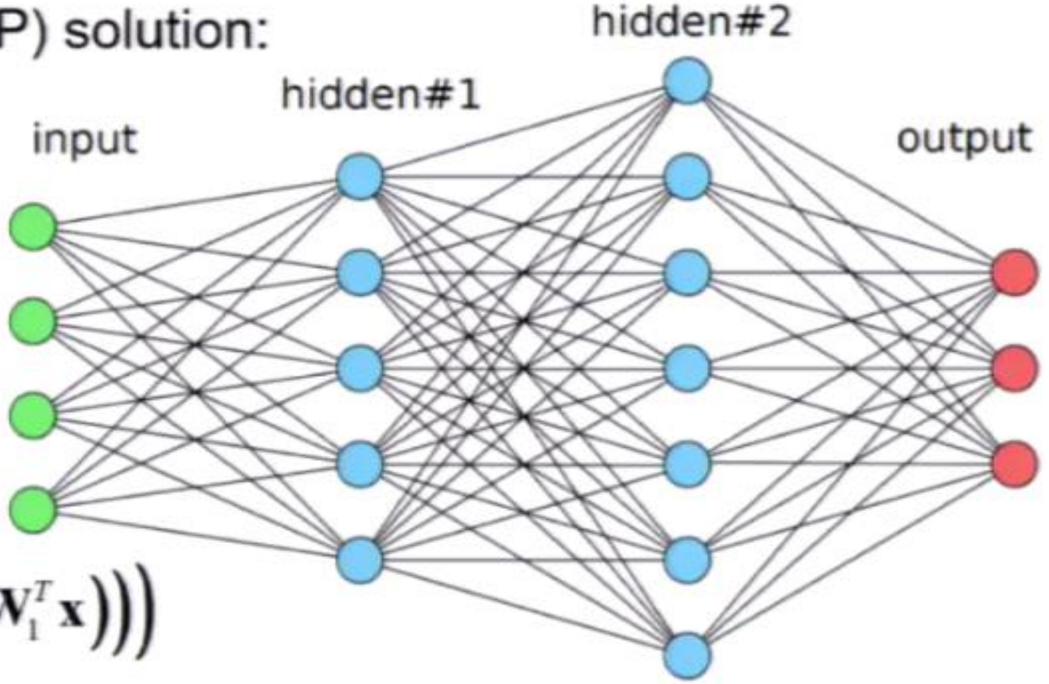


- Supervised learning: find the unknown mapping or function:

  $f(\mathrm{g})$: $\mathbf{y} = f(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{Z}^c$

  using discrete known examples $\{\mathbf{x}_i, \mathbf{y}_i\}$ for i=1, 2, ..., l.

- Neural network (MLP) solution:



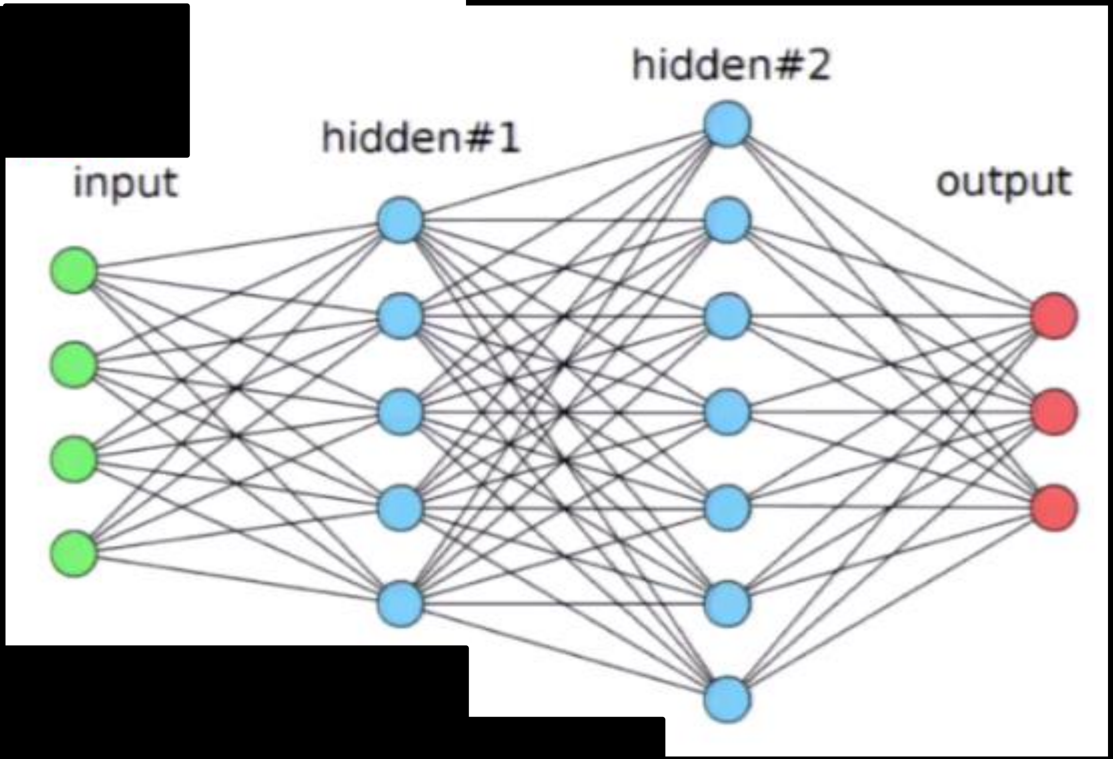
$$\mathbf{y} = h\left(\mathbf{W}_m^T \mathrm{L} \;\; h\left(\mathbf{W}_2^T h\left(\mathbf{W}_1^T \mathbf{x}\right)\right)\right)$$

$h(\mathrm{g})$: simple nonlinear function

Understanding Deep Learning

$$\mathbf{y} = h\left(\mathbf{W}_m^T \mathrm{L}\ h\left(\mathbf{W}_2^T h\left(\mathbf{W}_1^T \mathbf{x}\right)\right)\right) \quad \Rightarrow \quad \mathbf{y} = f(\mathbf{x})$$

- Theoretically, m=2 is sufficient to approximate any highly nonlinear function, i.e. $e \Rightarrow 0$



- Problems of machine learning:

$f(\bullet)$: $\mathbf{y} = f(\mathbf{x})$, $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{Z}^c$

using finite discrete known training samples $\{\mathbf{x}_i, \mathbf{y}_i\}$ for i=1, 2, ..., l.

This is to find: $\mathbf{y}_i = \hat{f}(\mathbf{x}_i)$ by $\min_{\hat{f}} e^2 = \min_{\hat{f}} \sum_{\forall i} \left\| \mathbf{y}_i - \hat{f}(\mathbf{x}_i) \right\|_2^2$

can only find: $\mathbf{y}_i = \hat{f}(\mathbf{x}_i)$ for i=1, 2, ..., l.

not $\mathbf{y} = f(\mathbf{x})$, for the wole population $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{Z}^c$

- How to make

$\hat{f} \Rightarrow f(\mathbf{x})$, for the wole population $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{Z}^c$?

- Regularization! Using human knowledge to restrict or constrain $\hat{f}$,

so that $\hat{f} \Rightarrow f(\mathbf{x})$, for the whole population $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{Z}^c$

➢ Regularization! Using human knowledge to restrict or constrain

How to regularize $\hat{f}$?

$$\min_{\hat{f}} e^2 = \min_{\hat{f}} \sum_{\forall i} \left\| \mathbf{y}_i - \hat{f}(\mathbf{x}_i) \right\|_2^2$$

$$\Downarrow$$

$$\min_{\hat{f} \in \Omega} e^2 = \min_{\hat{f} \in \Omega} \sum_{\forall i} \left\| \mathbf{y}_i - \hat{f}(\mathbf{x}_i) \right\|_2^2$$

$$\text{or } \min_{\hat{f}} \sum_{\forall i} \left\| \mathbf{y}_i - \hat{f}(\mathbf{W}\mathbf{x}_i) \right\|_2^2$$

$$\min_{\hat{f}} \left[\sum_{\forall i} \left\| \mathbf{y}_i - \hat{f}(\mathbf{x}_i) \right\|_2^2 + \lambda \Phi(\hat{f}) \right]$$

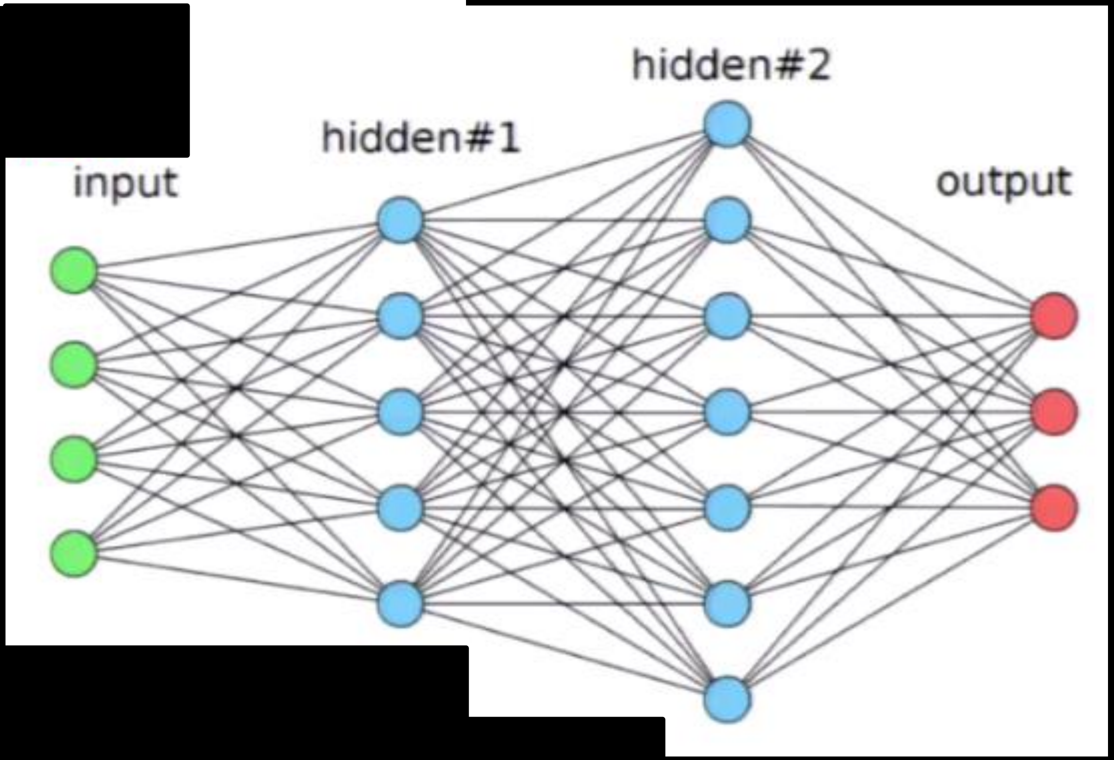
or all of the above

Understanding Deep Learning

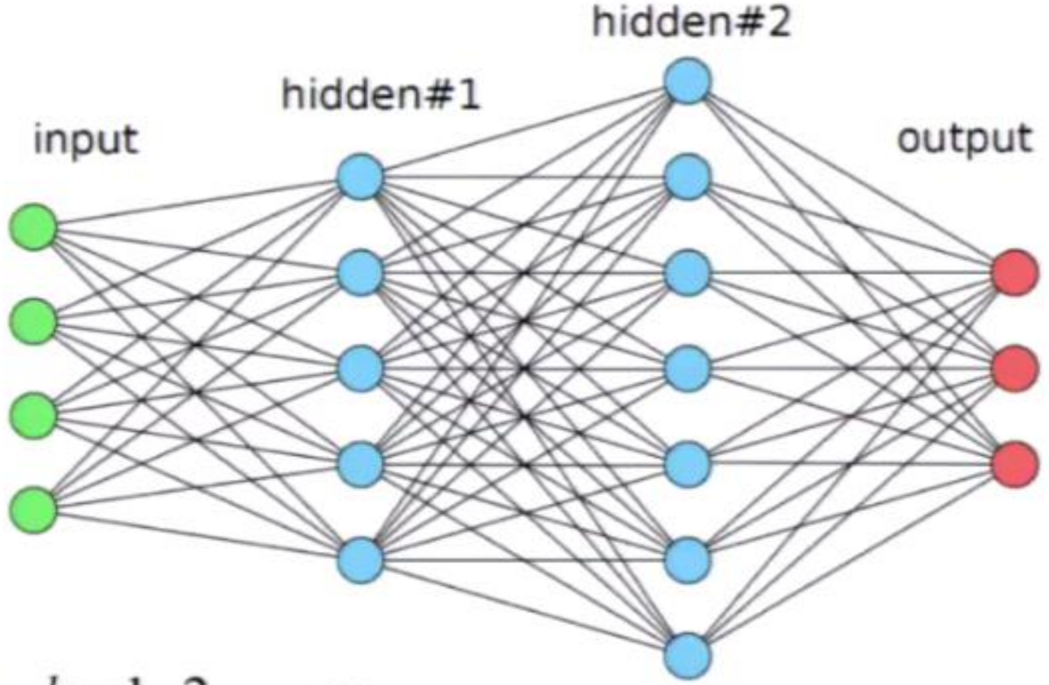
$$\mathbf{y} = h\left(\mathbf{W}_m^T \mathrm{L}\ h\left(\mathbf{W}_2^T h\left(\mathbf{W}_1^T \mathbf{x}\right)\right)\right) \Rightarrow \mathbf{y} = f(\mathbf{x})$$

- Theoretically, m=2 is sufficient to approximate any highly nonlinear function, i.e. $e \Rightarrow 0$

- Deep Learning m>>2.



$$\mathbf{y} = h\left(\mathbf{W}_m^T \mathrm{L}\ \ h\left(\mathbf{W}_2^T h\left(\mathbf{W}_1^T \mathbf{x}\right)\right)\right) \quad \Rightarrow \quad \mathbf{y} = f\left(\mathbf{x}\right)$$



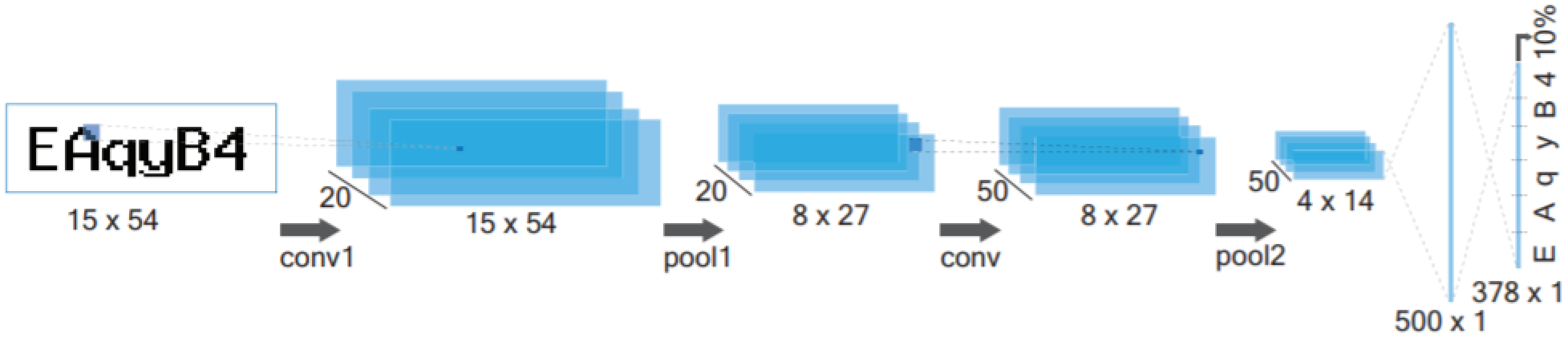
$$\mathbf{x}^{k+1} = h\left(\mathbf{W}_k^T \mathbf{x}^k\right), \quad k = 1,\ 2,\ \ldots,\ m$$

$$\mathbf{x}^1 = \mathbf{x}, \quad \mathbf{y} = \mathbf{x}^{m+1}, \quad \mathbf{o}^k = \mathbf{W}_k^T \mathbf{x}^k \Rightarrow \mathbf{o} = \mathbf{W}^T \mathbf{x}$$

$$\mathbf{x}^k \in \mathbf{R}^{n_k},\ \mathbf{o}^k, \mathbf{x}^{k+1} \in \mathbf{R}^{n_{k+1}},\ \mathbf{W}_k \in \mathbf{R}^{n_k \times n_{k+1}}$$

Convolutional Neural Networks

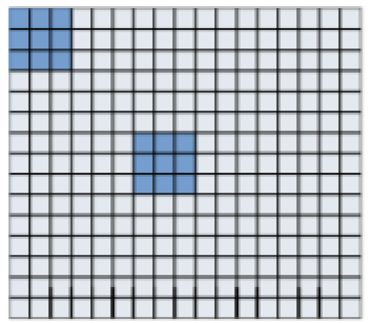
- CNNs are designed to process the data in the form of multiple arrays (e.g. 2D images, 3D video/volumetric images)
- Typical architecture is composed of series of stages: ***convolutional*** *layers* and ***pooling*** *layers*
- Each unit is connected to local patches in the feature maps of the previous layer



Key Idea behind Convolutional Networks

Convolutional networks take advantage of the properties of natural signals:

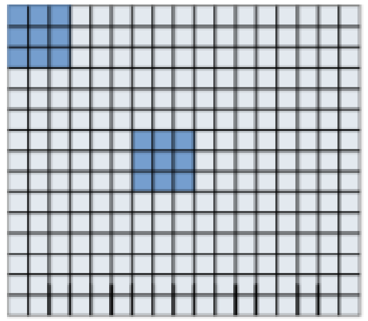
- local connections



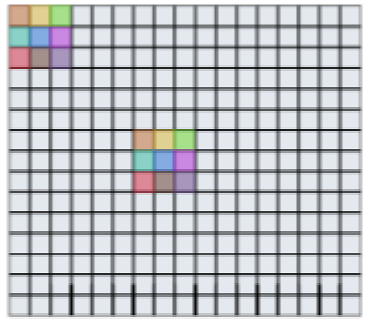
Key Idea behind Convolutional Networks

Convolutional networks take advantage of the properties of natural signals:

- local connections



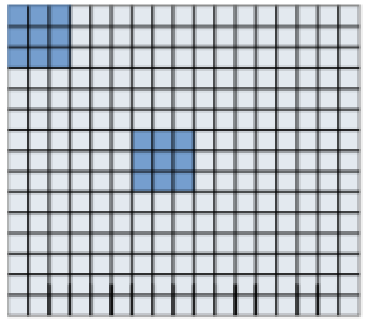
- shared weights



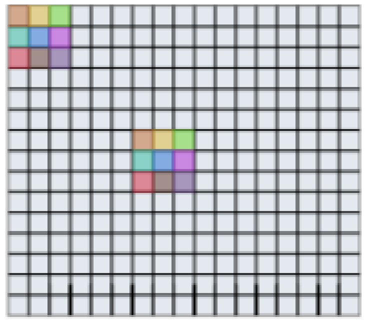
Key Idea behind Convolutional Networks

Convolutional networks take advantage of the properties of natural signals:

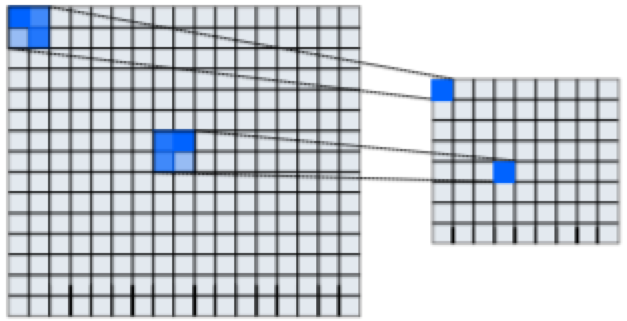
- local connections



- shared weights



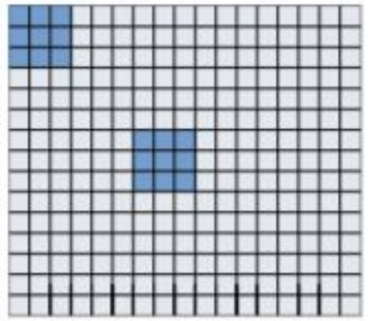
- pooling



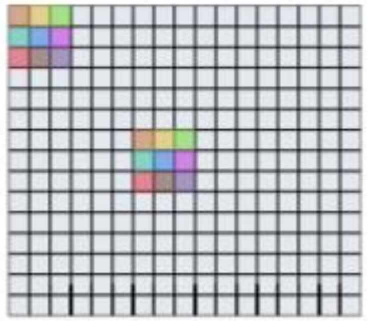
Key Idea behind Convolutional Networks

Convolutional networks take advantage of the properties of natural signals:

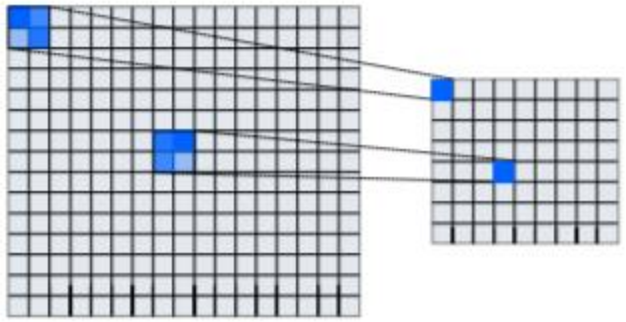
- local connections



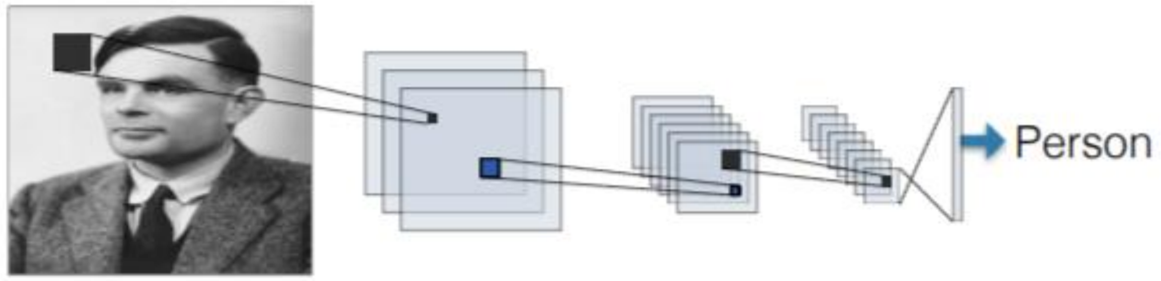
- shared weights



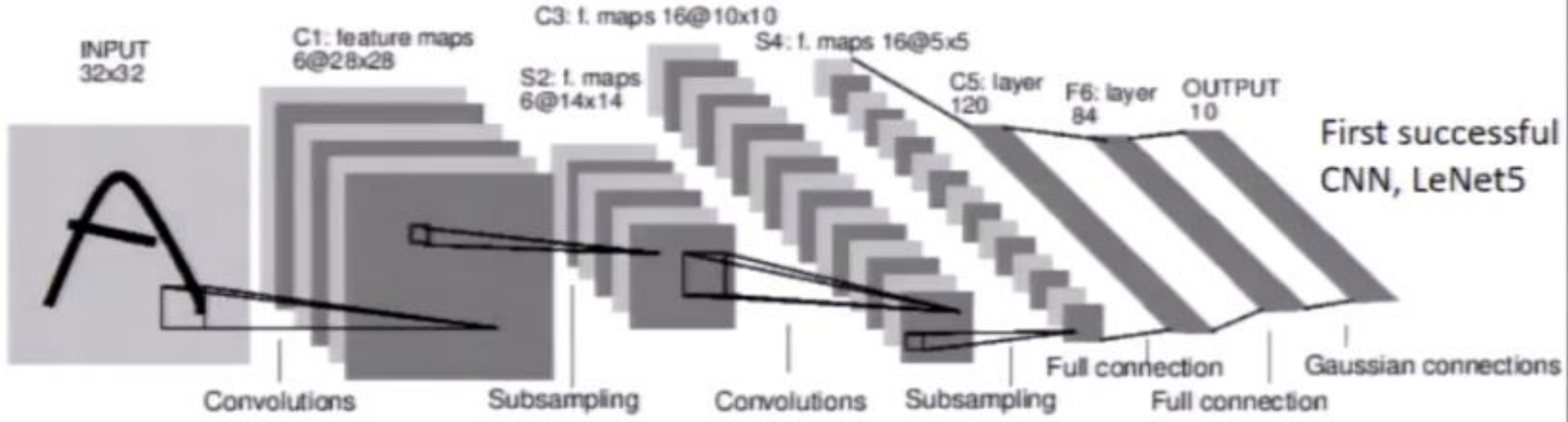
- pooling



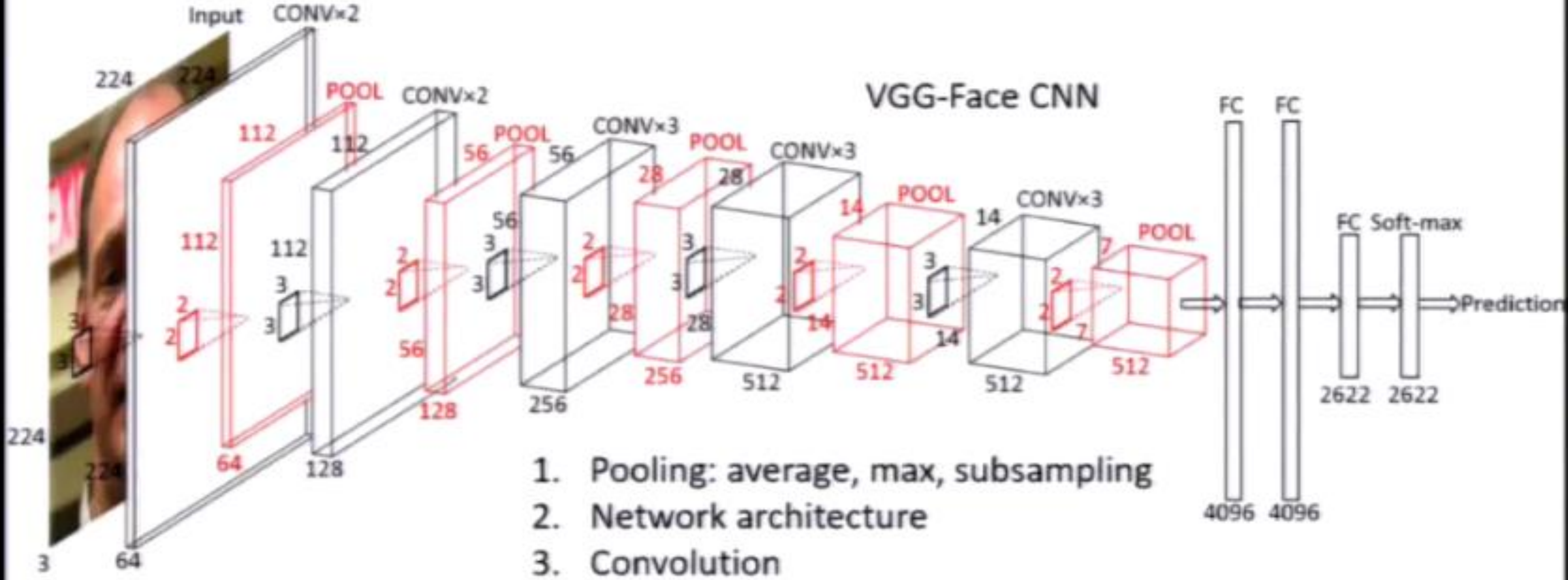
- the use of many layers



Convolutional network CNN appears to be quite different from MLP?



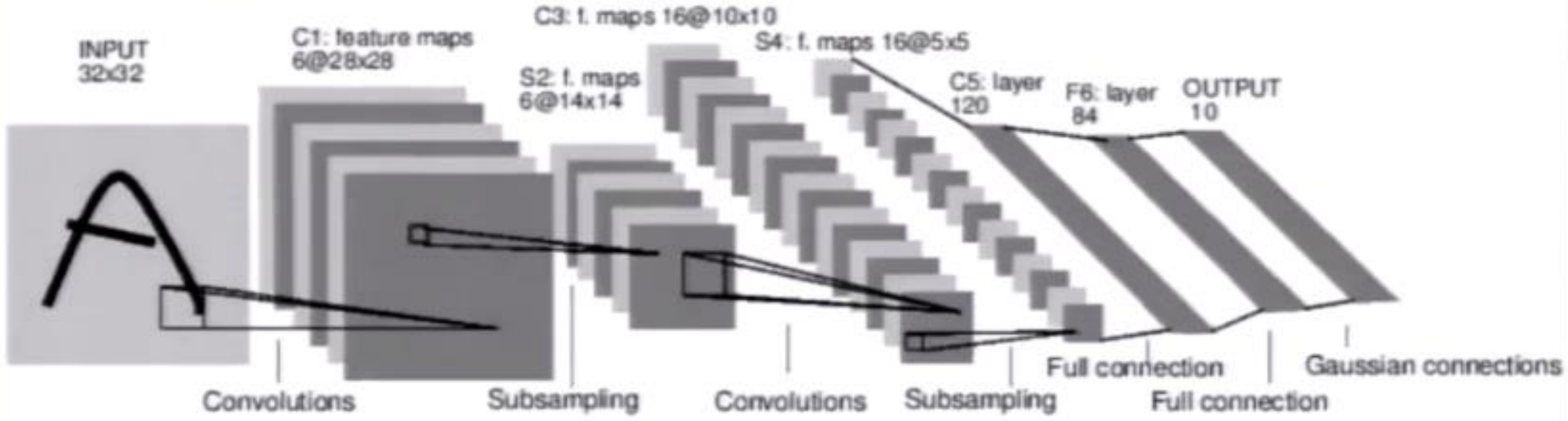
First successful CNN, LeNet5

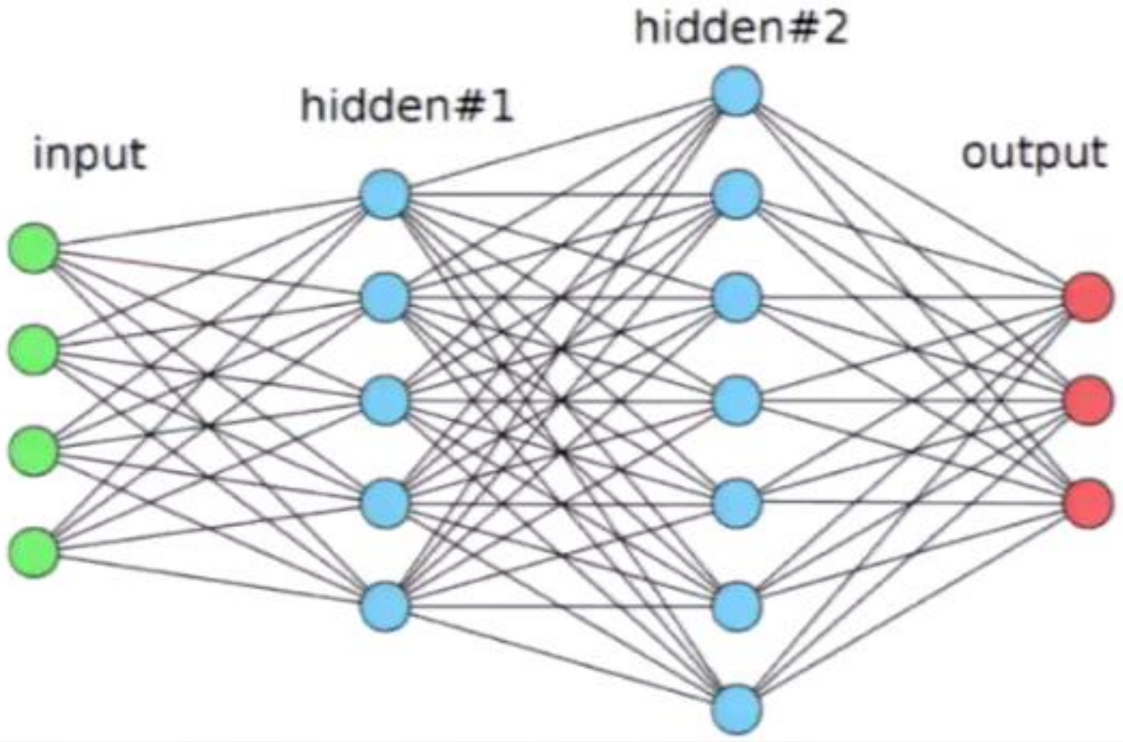


VGG-Face CNN

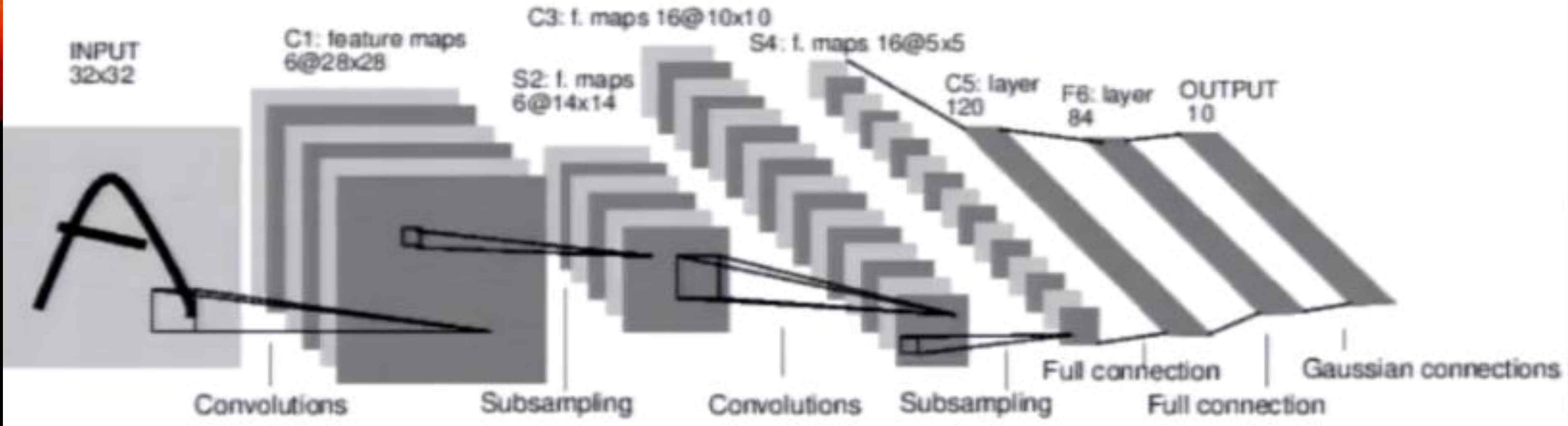
1. Pooling: average, max, subsampling
2. Network architecture
3. Convolution

Compare MLP and CNN: Network architecture





Compare MLP and CNN: Network architecture



$$\mathbf{o} = \mathbf{W}^T \mathbf{x}$$

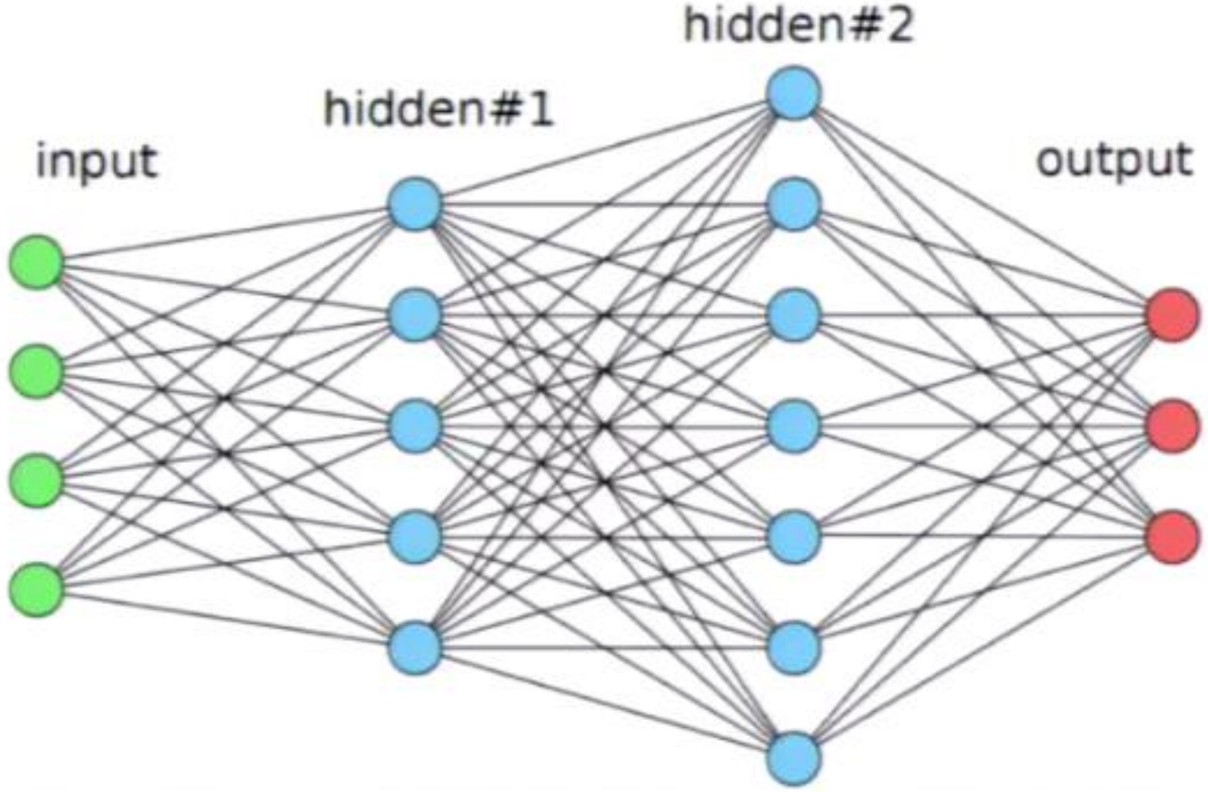
$$\mathbf{x} = \{x_j\} \in \mathbf{R}^{n_k}$$

$$\mathbf{o} = \{o_j\} \in \mathbf{R}^{n_{k+1}},$$

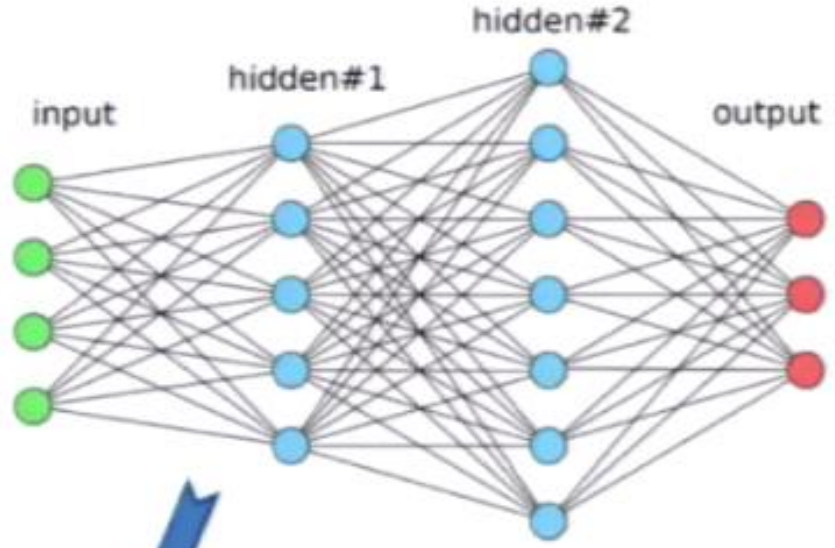
$$\mathbf{W} \in \mathbf{R}^{n_k \times n_{k+1}}$$

$$n_k = 32 \times 32 = 1024$$

$$n_{k+1} = 6 \times 28 \times 28 = 4704$$



Compare MLP and CNN:



$$\mathbf{W} = (\mathbf{W}^1, \ldots, \mathbf{W}^q \ldots, \mathbf{W}^p)$$

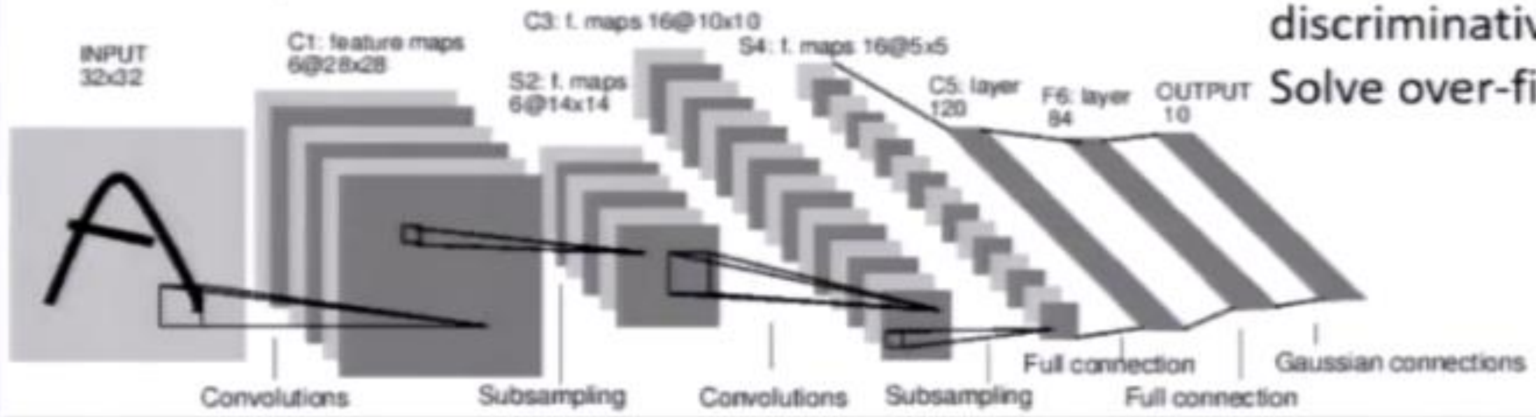
$$= \begin{pmatrix} \mathbf{g}^1 & 0 & 0 & & \mathbf{g}^p & 0 & 0 \\ M & M & M & & M & M & M \\ 0 & 0 & 0 & & 0 & 0 & 0 \\ 0 \ldots & \mathbf{g}^1 \ldots & 0 & \ldots\ldots & 0 \ldots & \mathbf{g}^p \ldots & 0 \\ 0 & 0 & 0 & & 0 & 0 & 0 \\ M & M & M & & M & M & M \\ 0 & 0 & \mathbf{g}^1 & & 0 & 0 & \mathbf{g}^p \end{pmatrix}$$

$$\mathbf{o} = \mathbf{W}^T \mathbf{x}$$

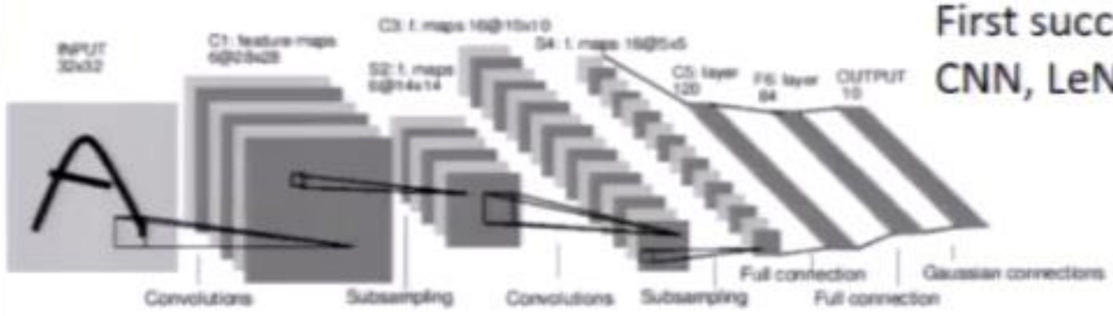
CNN is a simplified MLP

CNN is a regularized MLP

Simplification/regularization extracts less amount of discriminative information, Solve over-fitting problem.

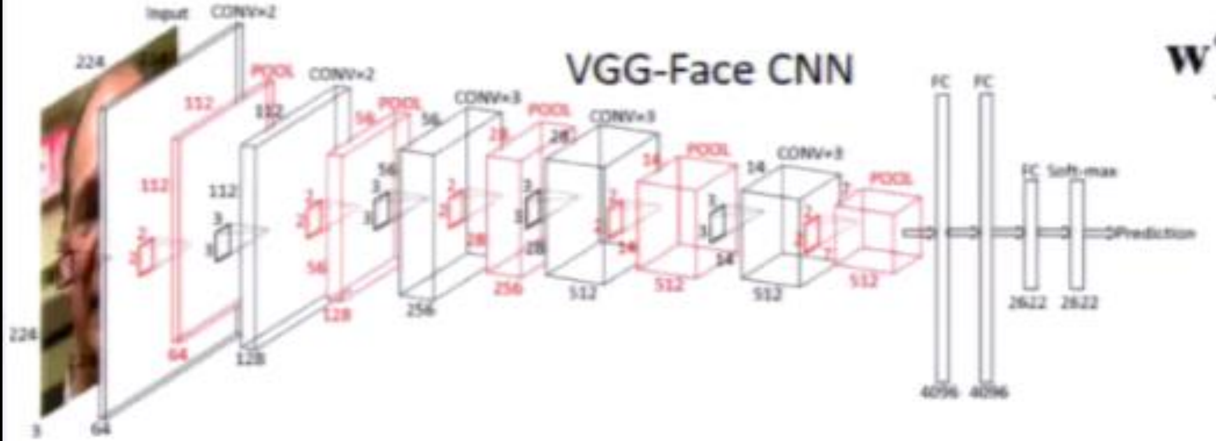


Merits of convolutional network, CNN



First successful CNN, LeNet5

VGG-Face CNN



$$\mathbf{w}_j^q = \begin{pmatrix} 0 \\ \mathrm{M} \\ 0 \\ \mathbf{g}^q \\ 0 \\ \mathrm{M} \\ 0 \end{pmatrix}$$

$$o_j^q = g_j^q * x_j$$

$$= \sum_{i=1}^{n_k} g_{j-i}^q * x_i$$

$$= \mathbf{w}_j^{qT} \mathbf{x}$$

1. Small filter size, forcing all other weights zero, captures image local structure / pattern.

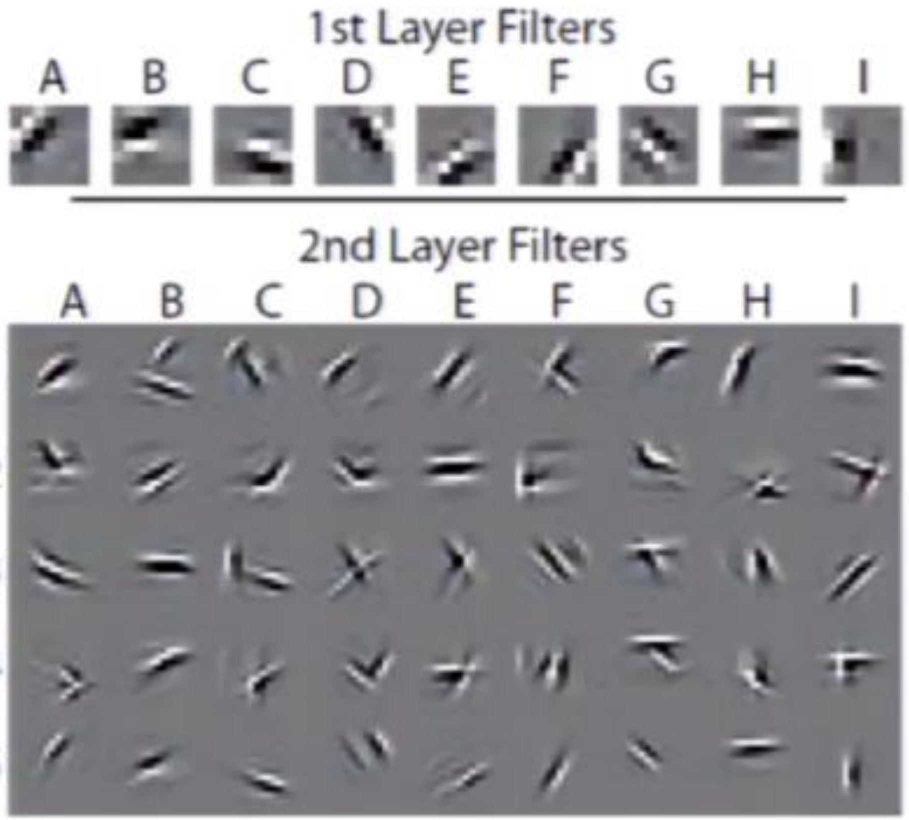
2. The convolution kernel, filter, is an image.

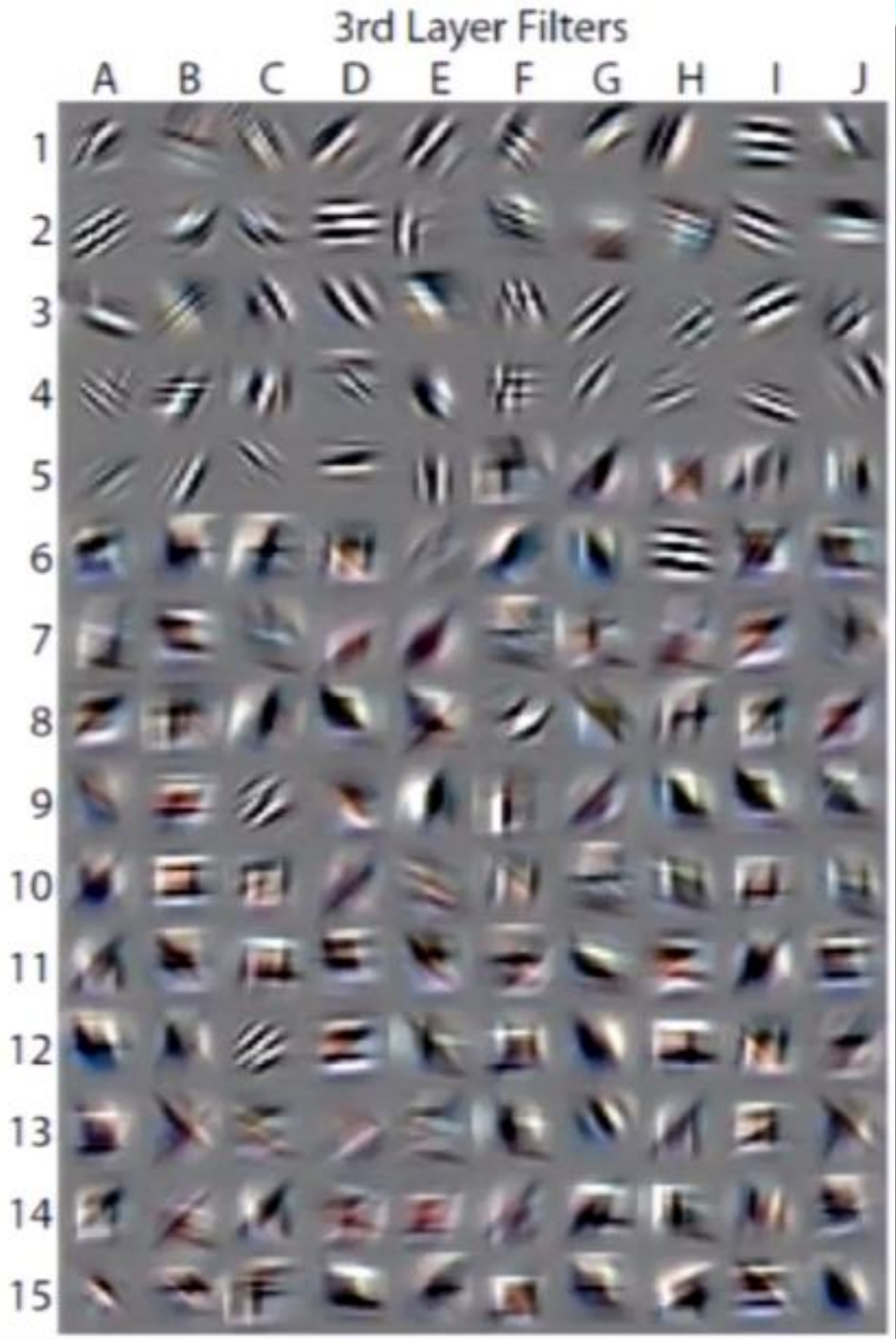
3. Convolution process is the same as correlation processing, matched filter.

4. A input image patch similar to the filter mask produces high output while those dissimilar to the filter mask produce low outputs.

5. A filter is trained to extract a local image structure, blob, corner, line, edge, curve, etc.

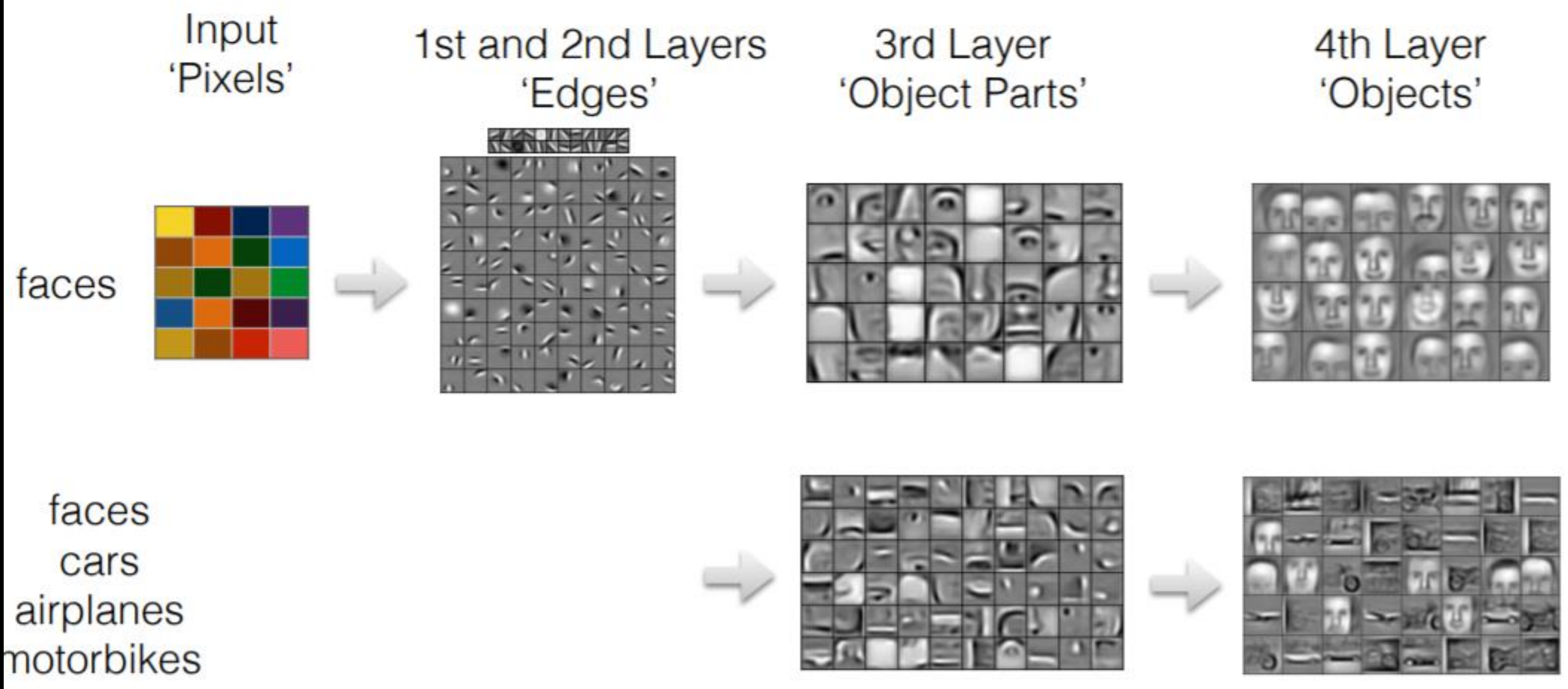
Further Examples:



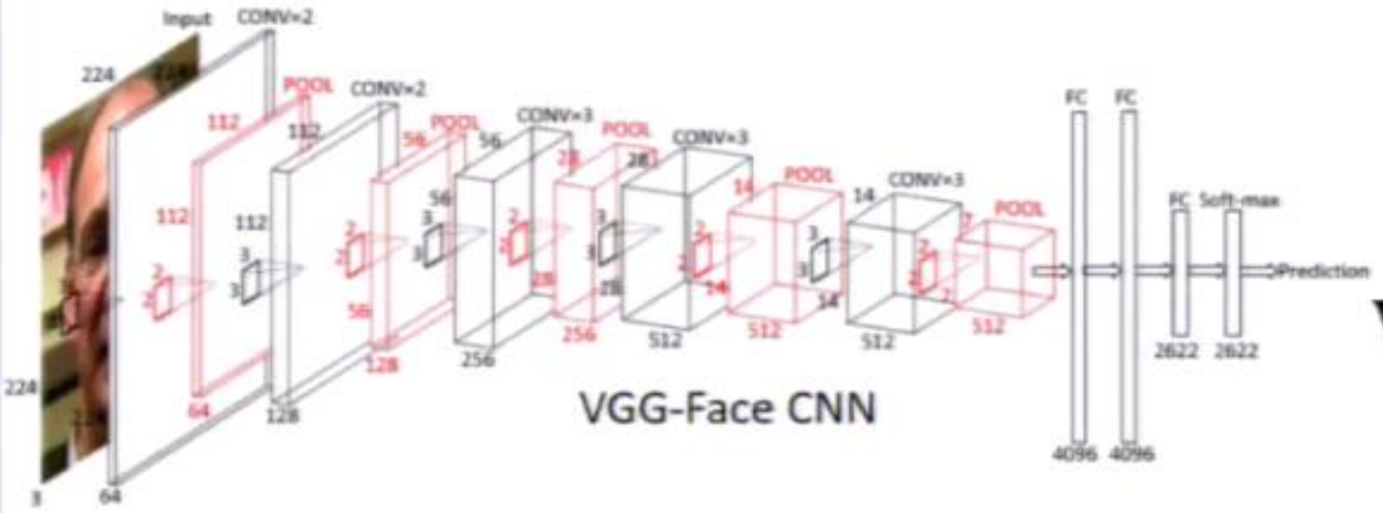


Filters trained on food scenes. Note the rich diversity of filters and their increasing complexity with each layer. In contrast to the filters shown in previous slide, the filters are evenly distributed over orientation.

Going deeper in the network



Further merits of convolutional network, CNN



$$\mathbf{W}^q = \begin{pmatrix} \mathbf{g}^q & 0 & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{M} \\ 0 & 0 & 0 \\ 0 \ldots & \mathbf{g}^q & \ldots 0 \\ 0 & 0 & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{M} \\ 0 & 0 & \mathbf{g}^q \end{pmatrix}$$

$$\mathbf{o}^q = \mathbf{W}^{q^T}\mathbf{x}$$

6. Similar image local structures may appear at many different locations of image. Convolution process of one filter extracts one image local structure at all locations

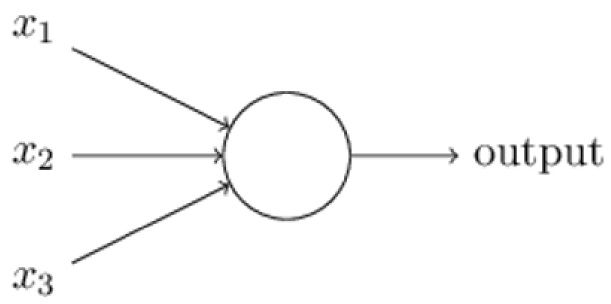
7. Multiple filters extract multiple image local structures at all locations.

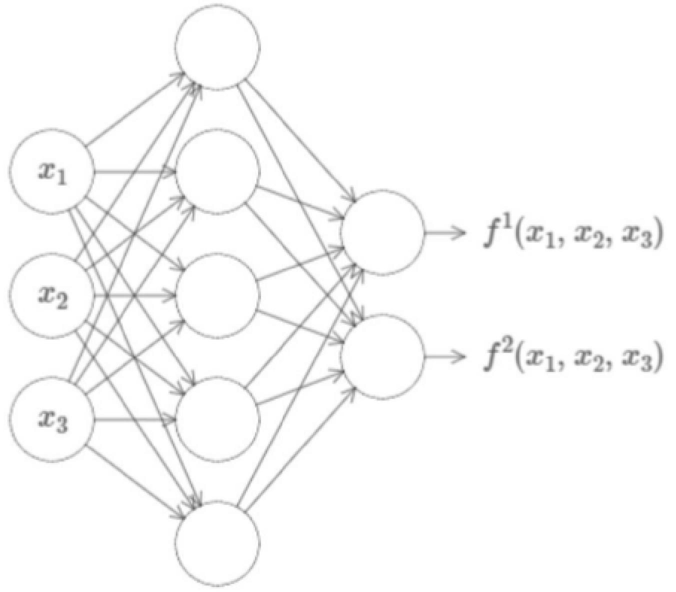
8. Pooling process reduces the image size, scale, so that the same filter size at higher layer captures larger scale image local structure.

9. Why deep network with many layers? image structure/pattern has different scales; Solving complex problem gradually one by one.

10. Even if one layer is ineffective or totally useless, no problem so long as it does not lose information. We have many next layers!!!

Combing Neurons In Hidden Layers

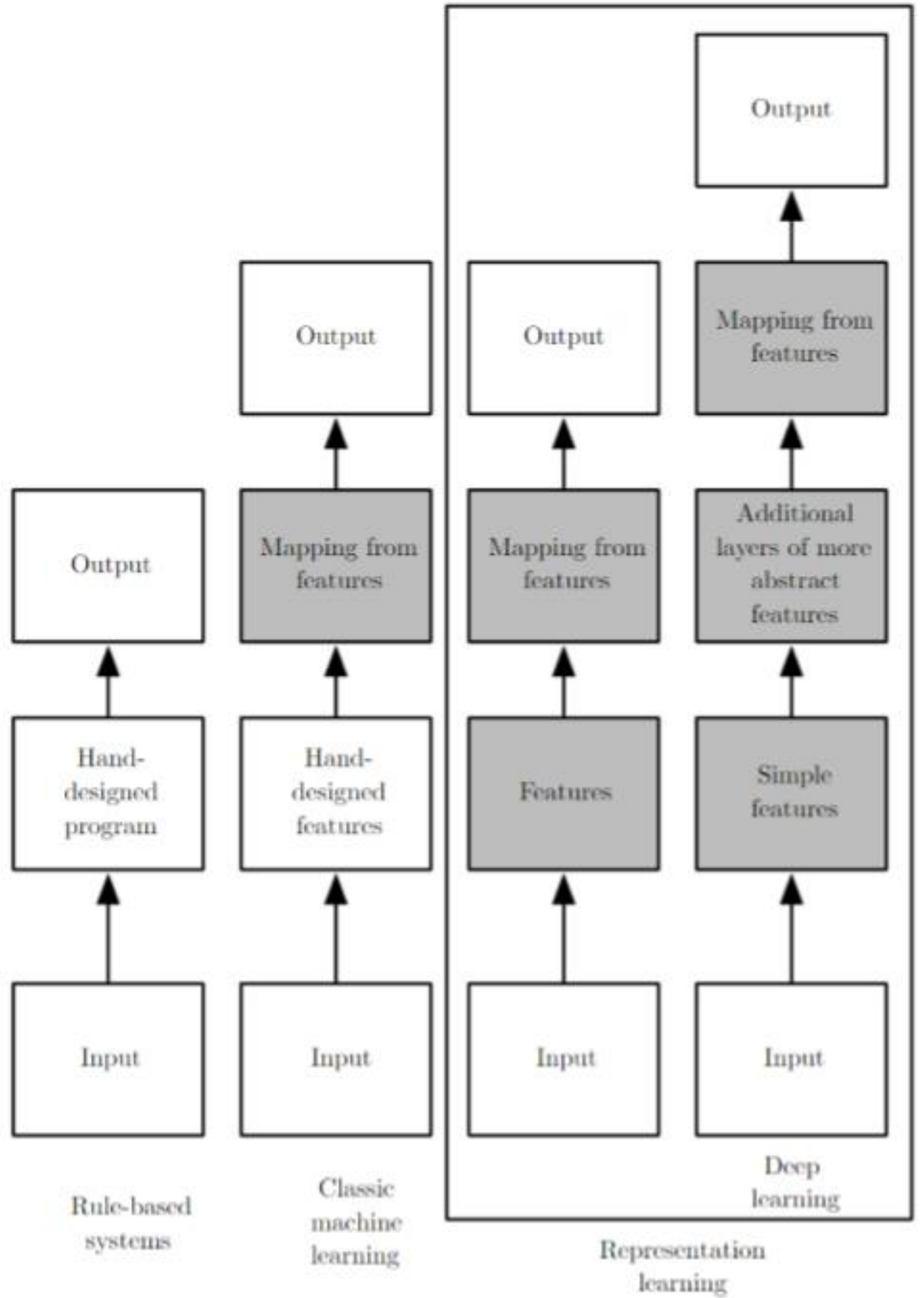




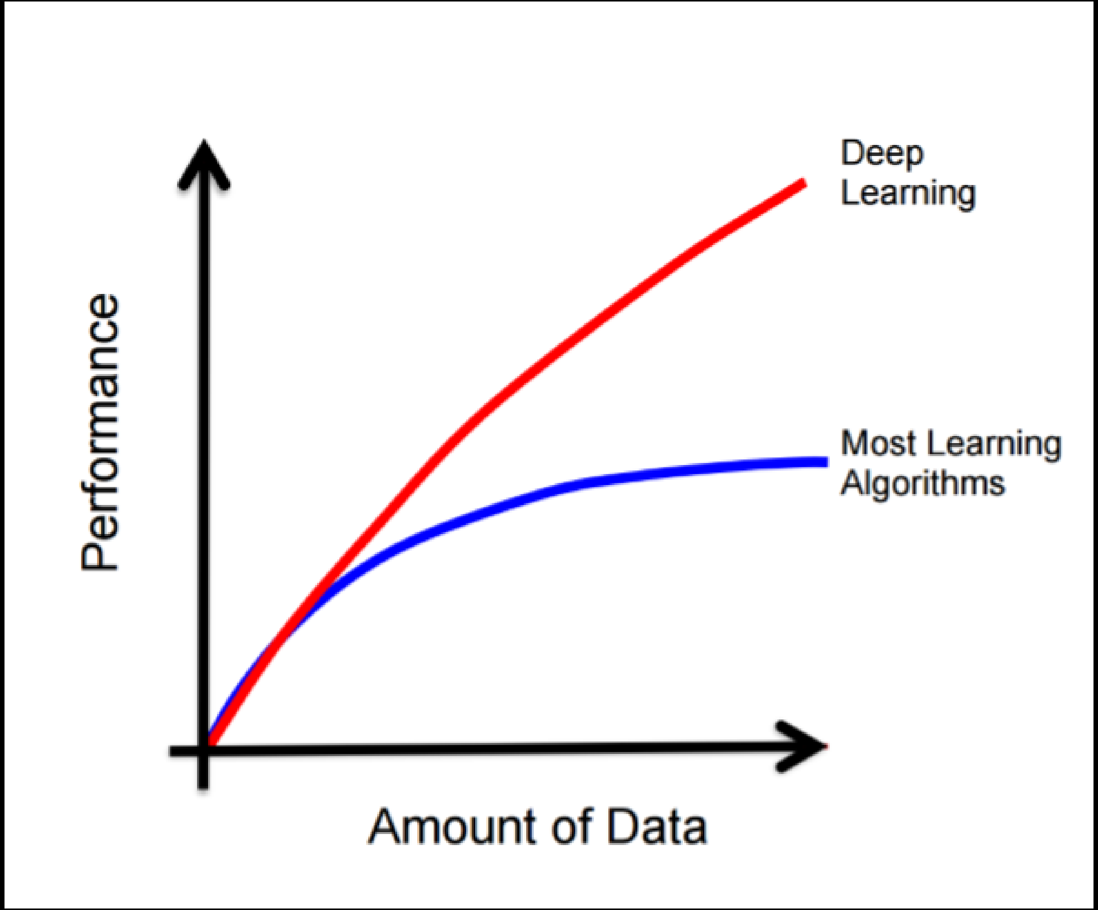
Universality: For any arbitrary function f(x), there exists a neural network that closely approximate it for any input x

Universality is an incredible property!* And it holds for just 1 hidden layer.

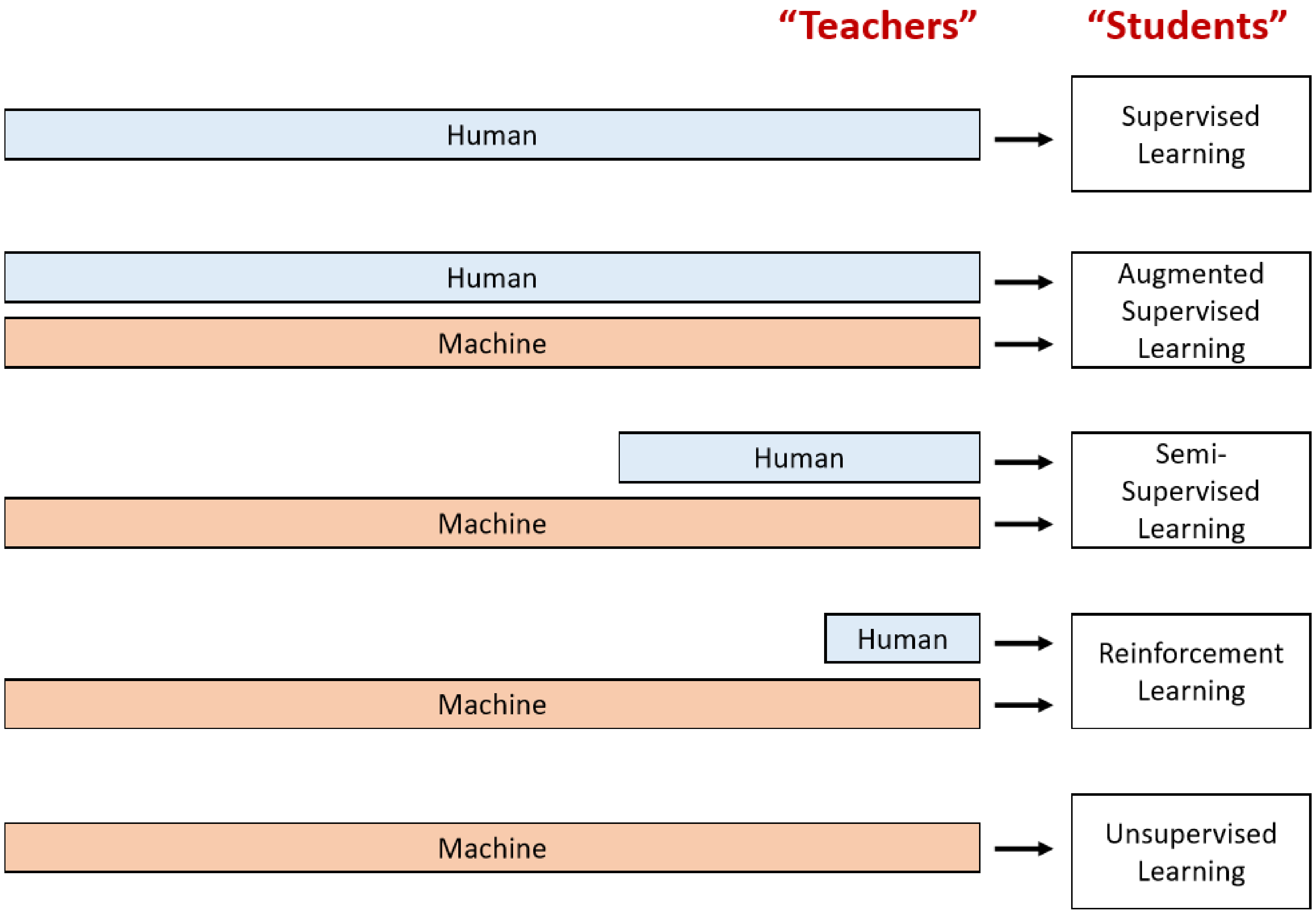
* Given that we have good algorithms for training these networks.



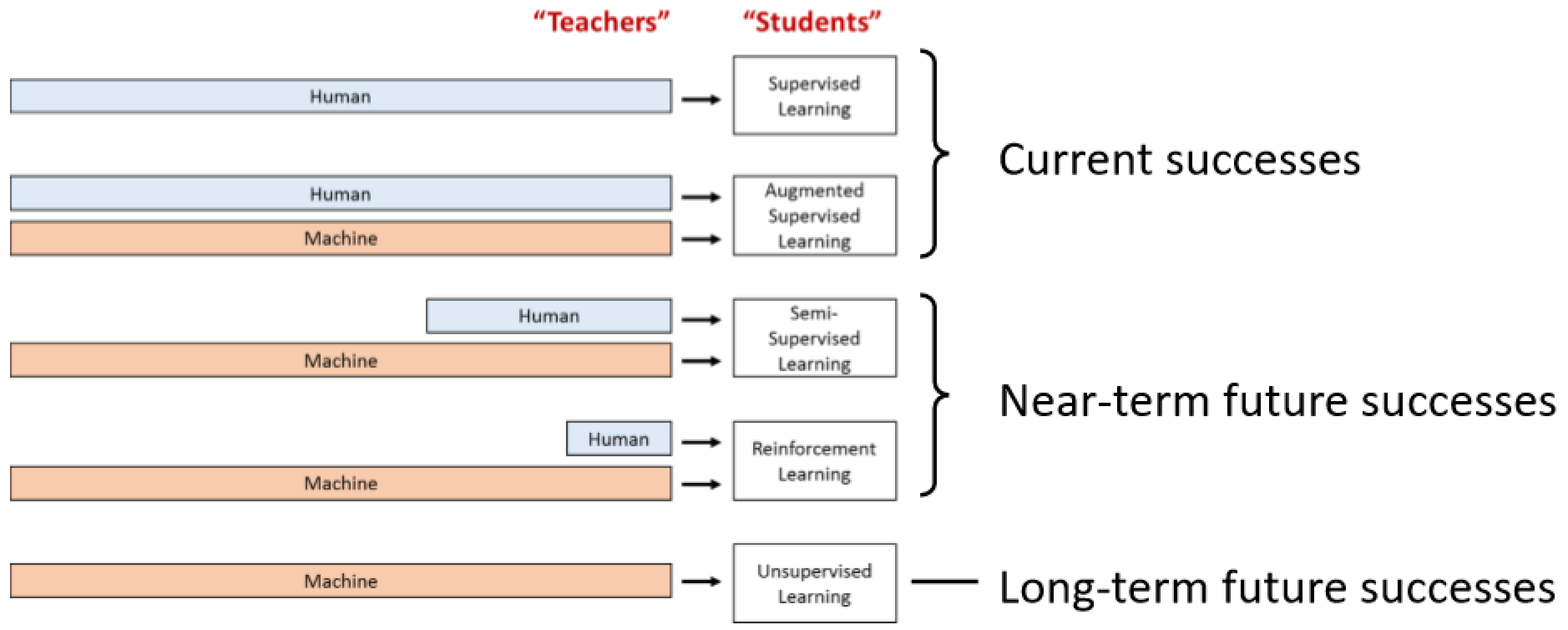
Scalable Machine Learning



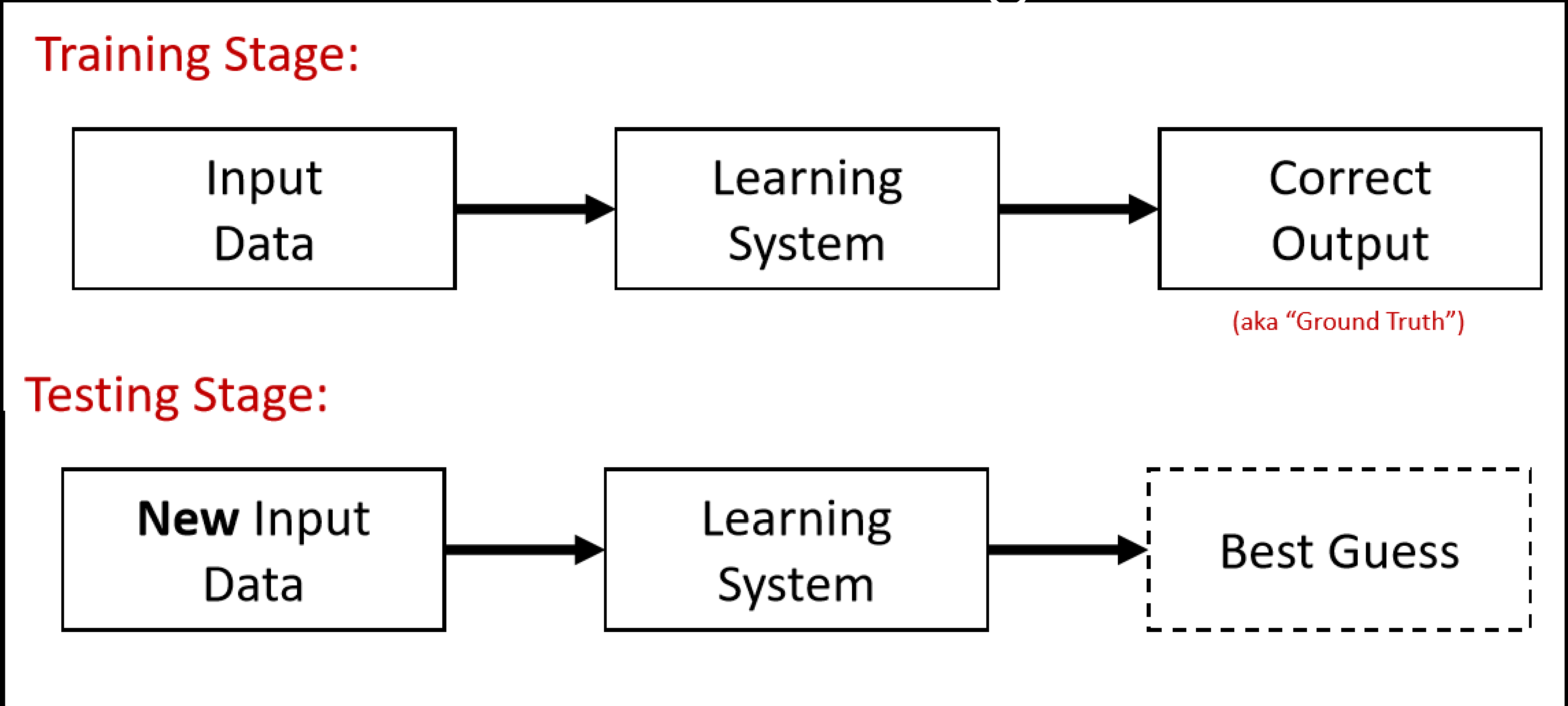
Deep Learning from Human and Machine



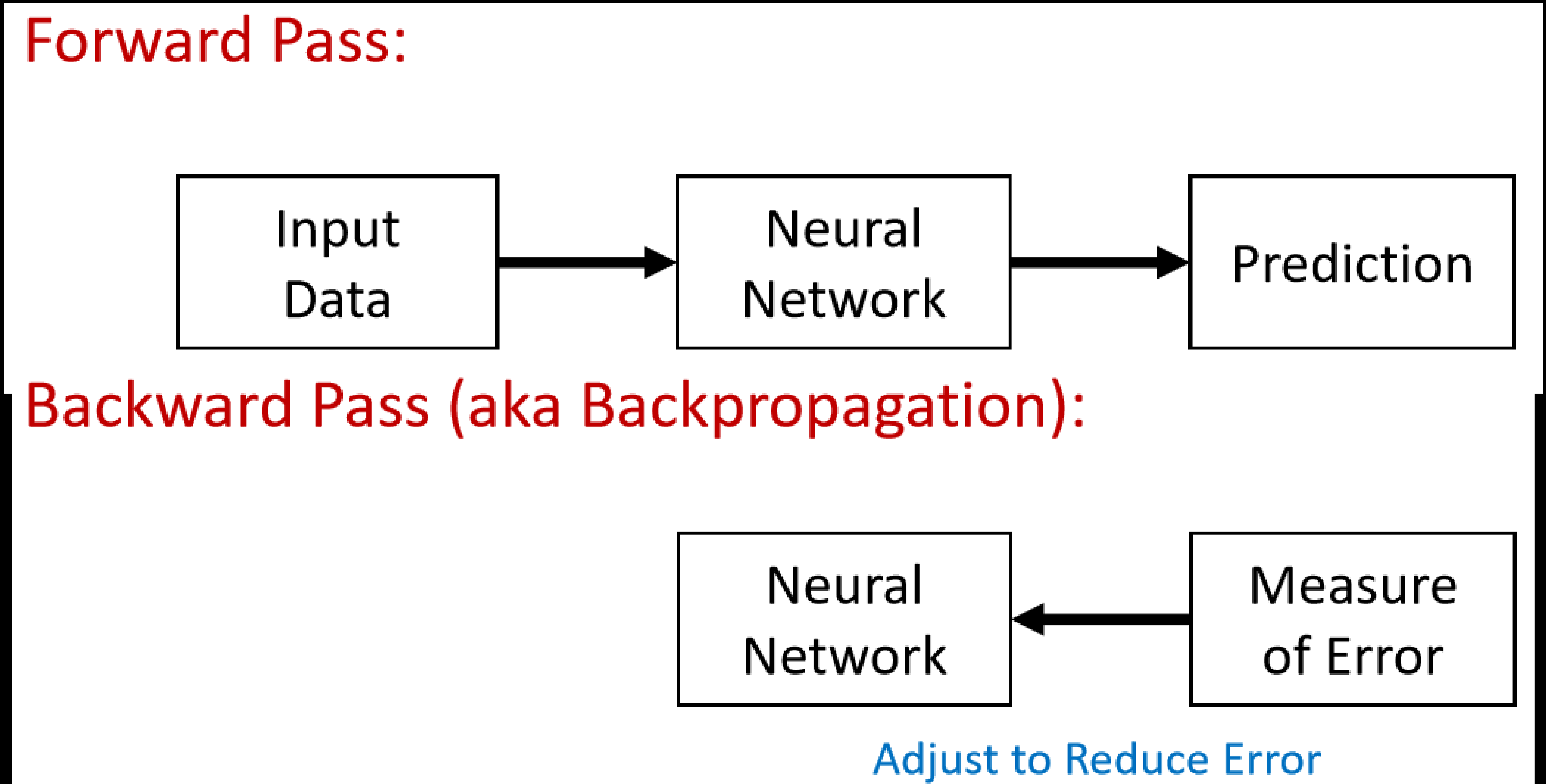
Deep Learning from Human and Machine



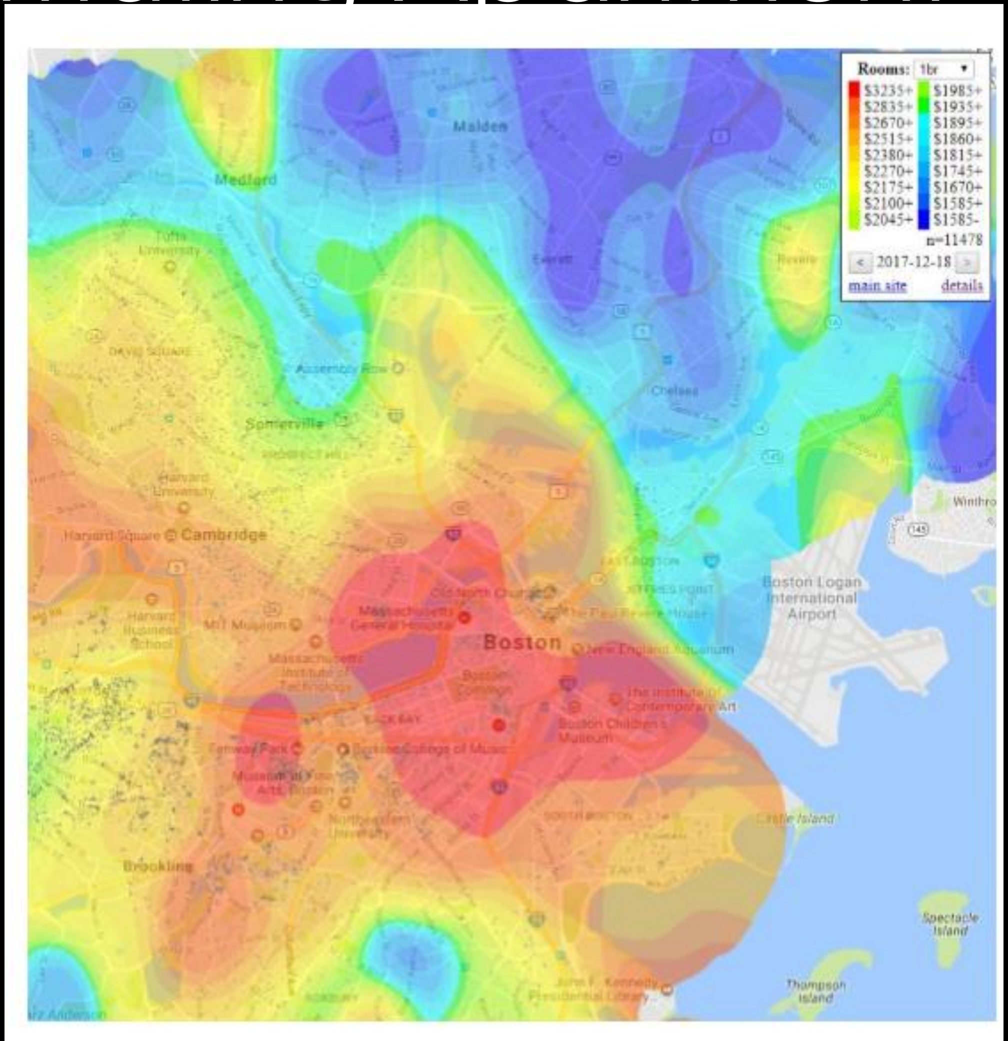
Deep Learning: Training And Testing



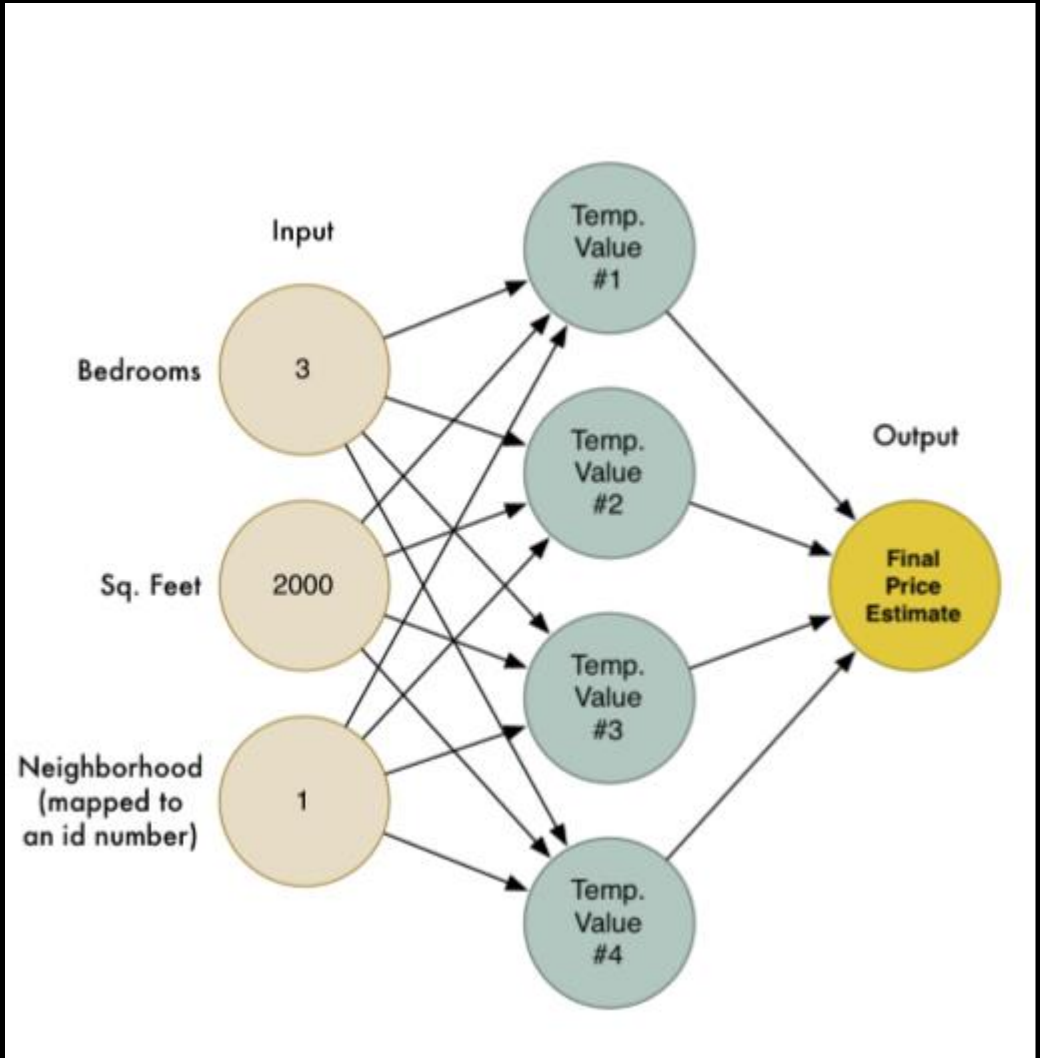
How Neural Networks Learn: Backpropagation



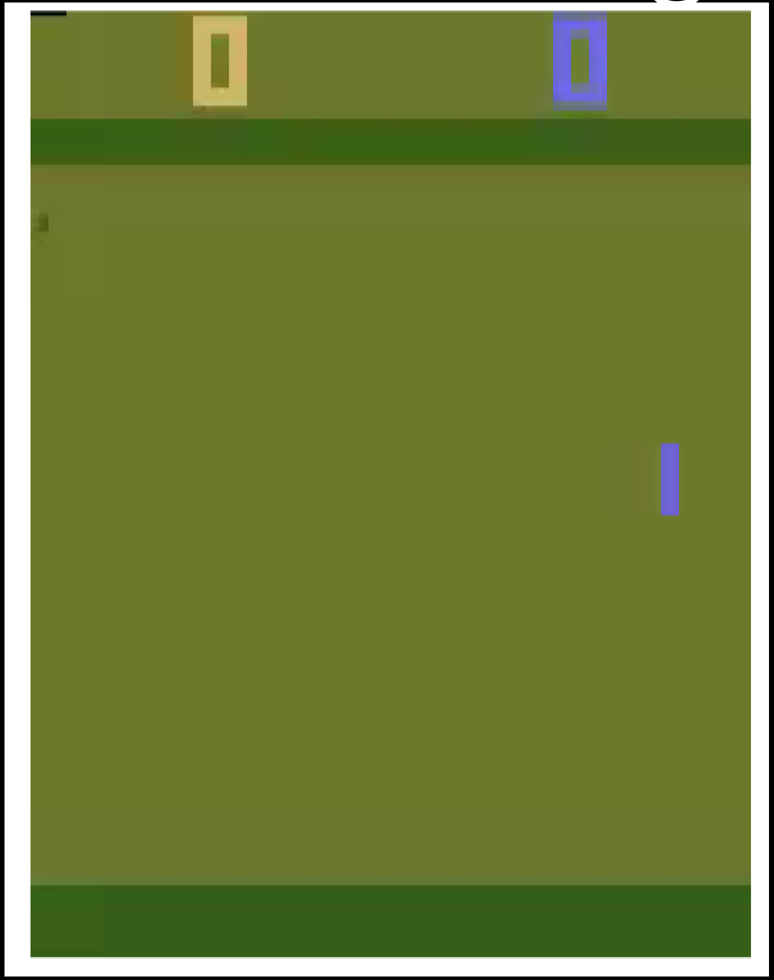
Special Purpose Intelligence: Estimating Apartment Cost



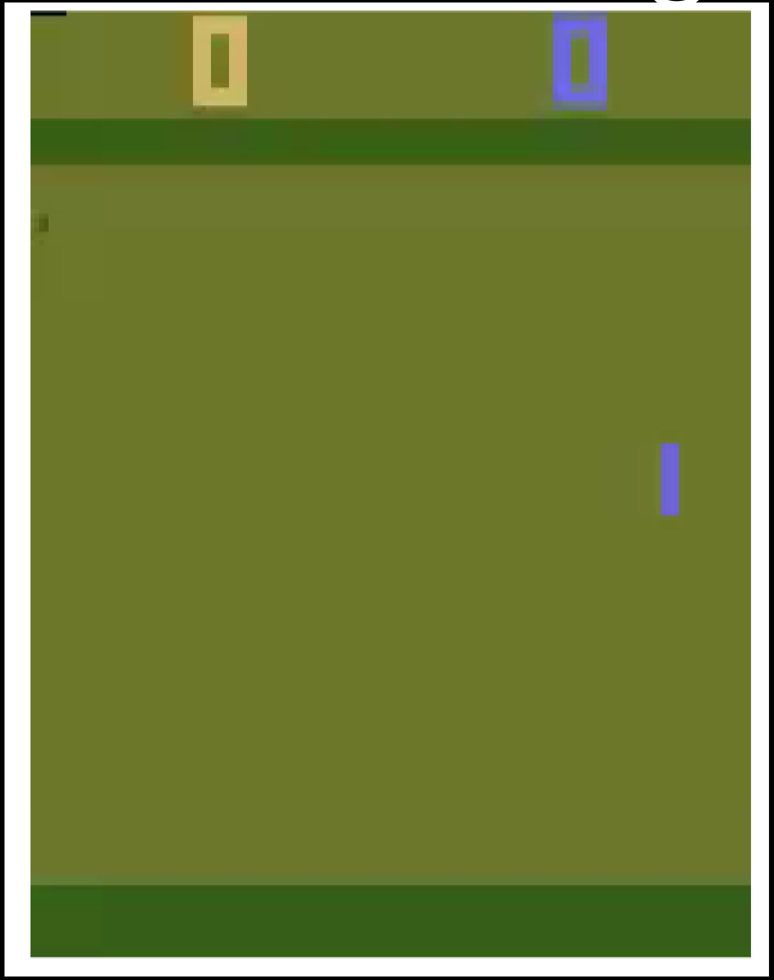
Estimating Apartment Cost

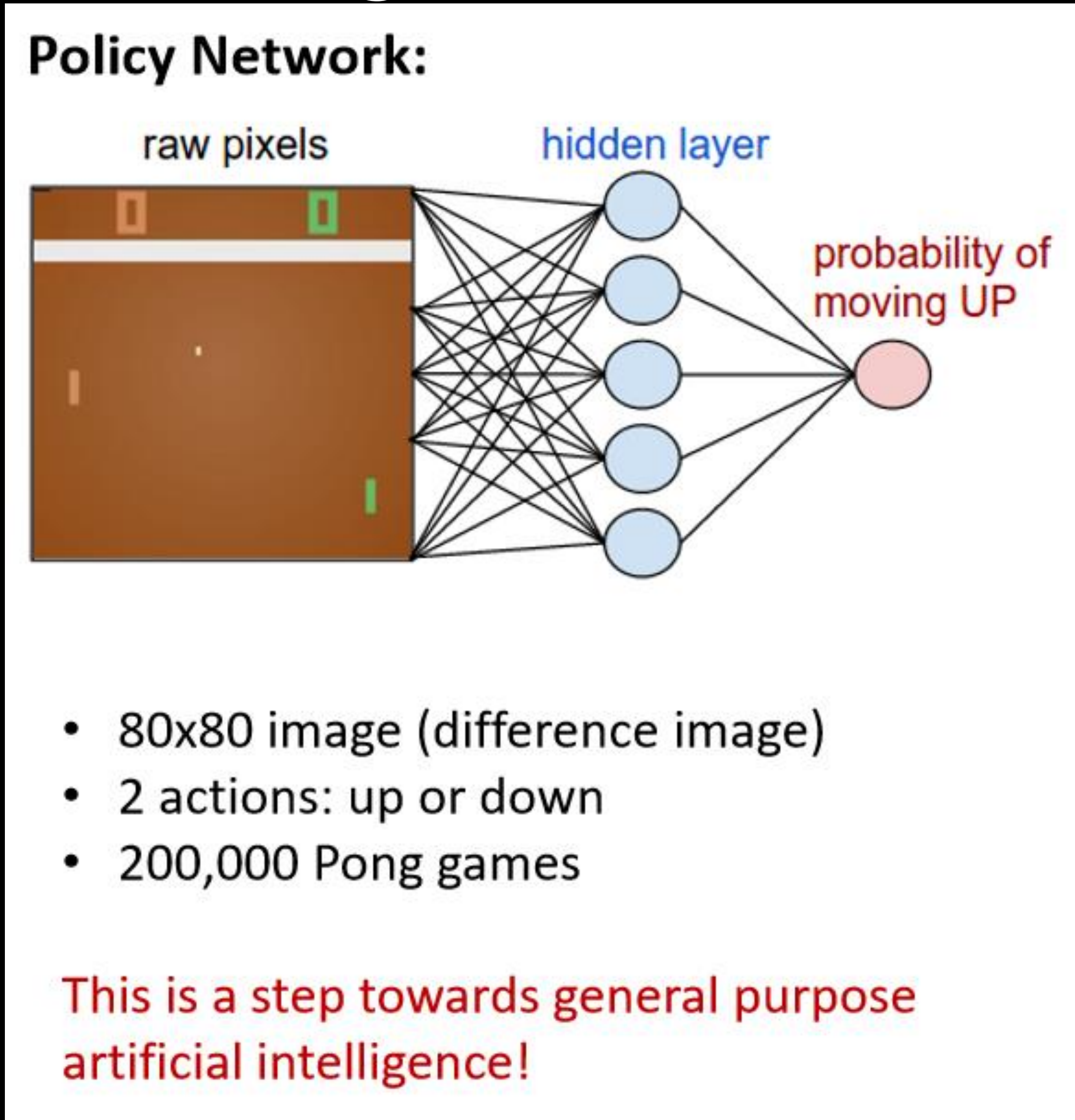


General Purpose Intelligence: Pong to Pixels



General Purpose Intelligence: Pong to Pixels





Policy Network:

- 80x80 image (difference image)
- 2 actions: up or down
- 200,000 Pong games

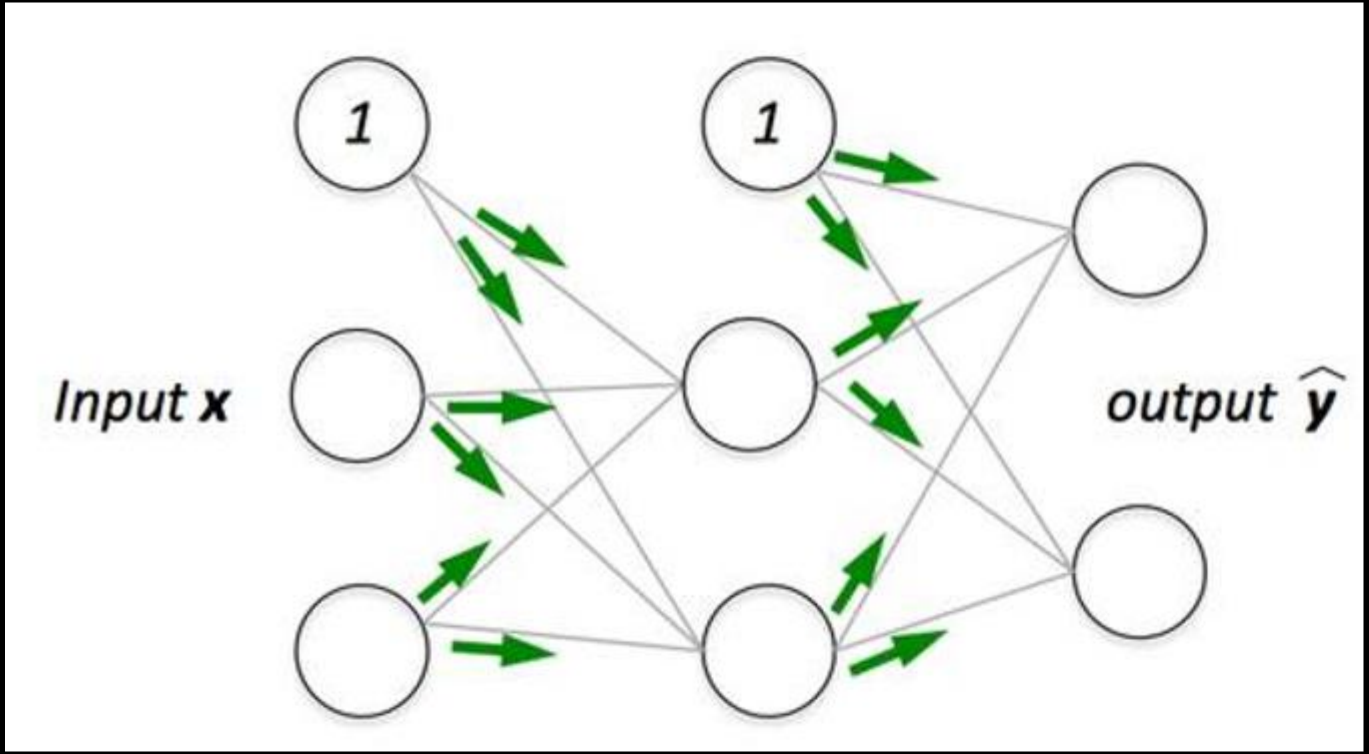
This is a step towards general purpose artificial intelligence!

Backpropagation

Update the **weights** and biases to decrease **loss function**

Loss function:

$$C = \frac{(y - a)^2}{2}$$

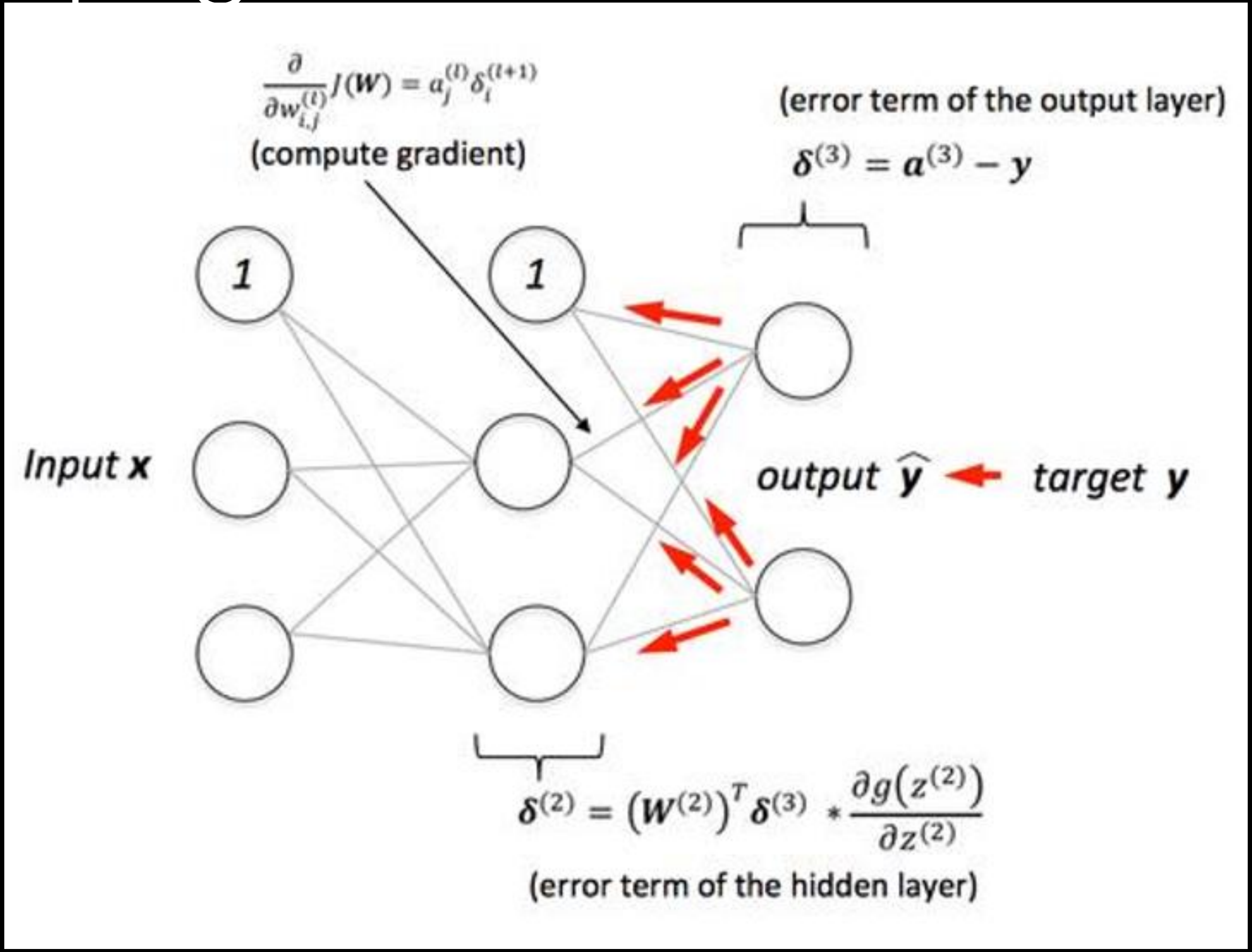


Forward pass to compute network output and "error"

Backward pass to compute gradients

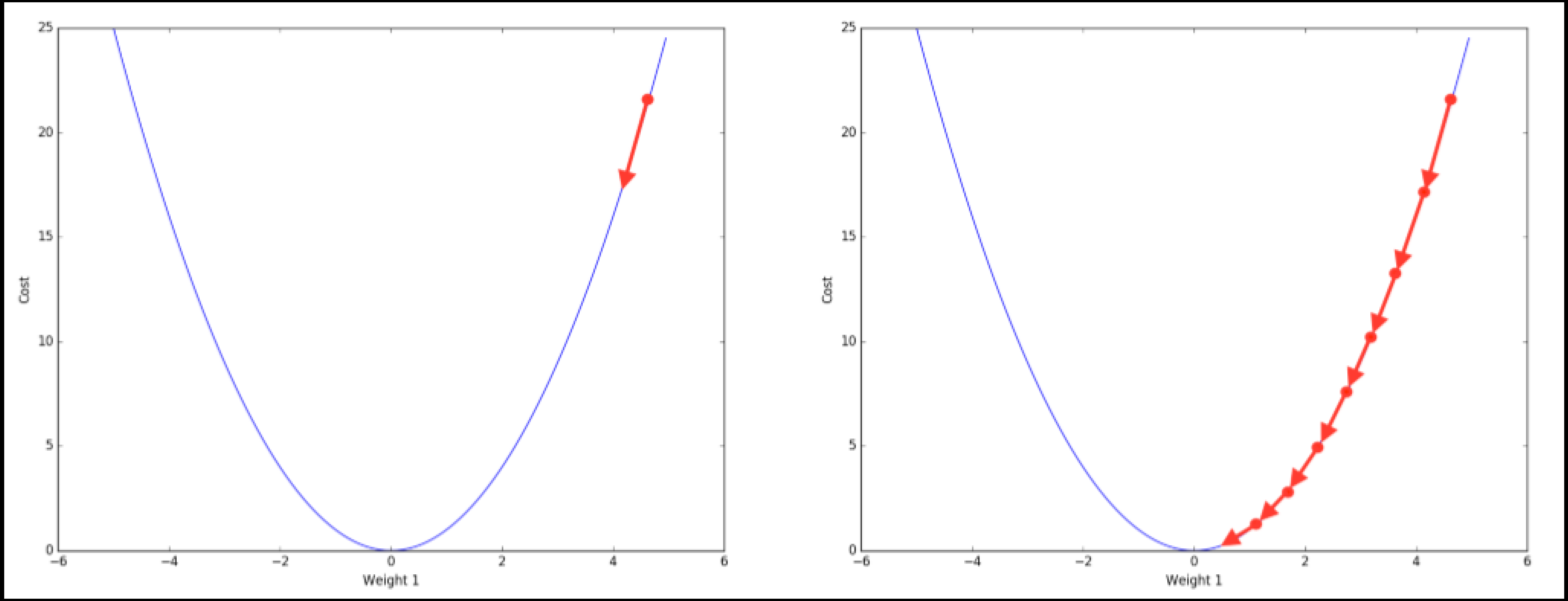
A fraction of the weight's gradient is subtracted from the weight.

Backpropagation

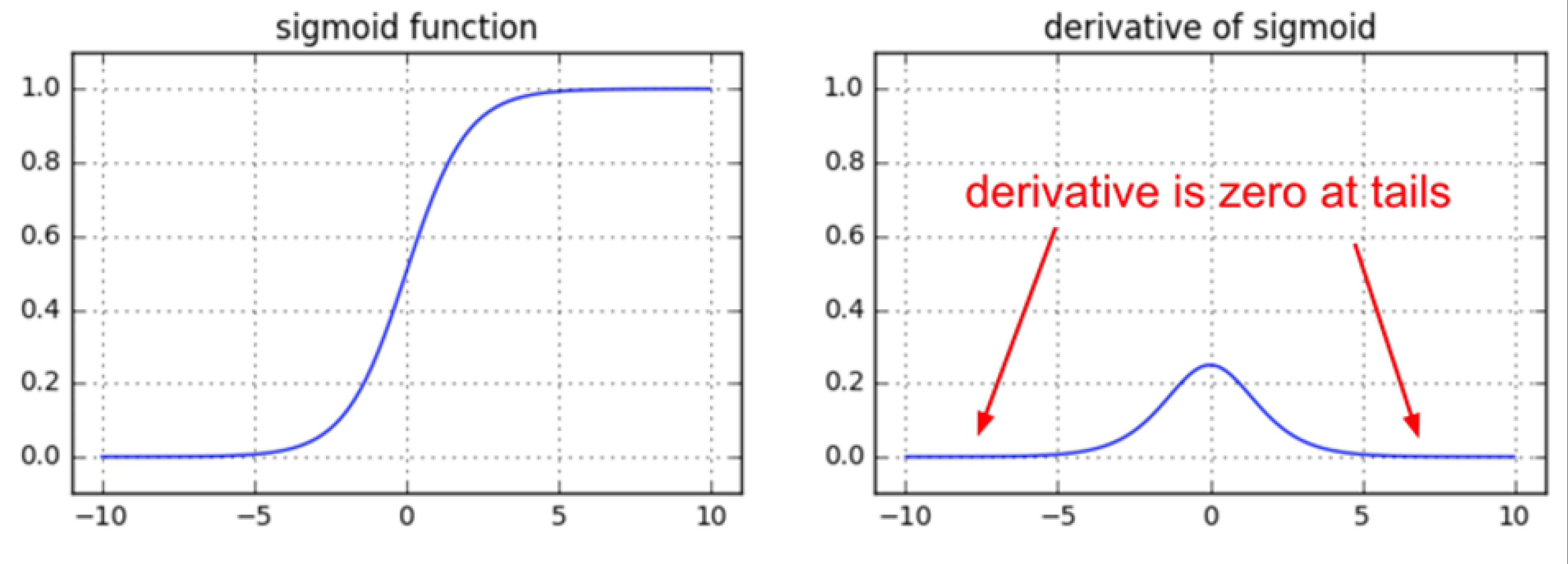

$\frac{\partial}{\partial w_{i,j}^{(l)}} J(W) = a_j^{(l)} \delta_i^{(l+1)}$
(compute gradient)
(error term of the output layer)
$\delta^{(3)} = a^{(3)} - y$
1
1
Input x
output $\hat{y}$
target y
$\delta^{(2)} = (W^{(2)})^T \delta^{(3)} * \frac{\partial g(z^{(2)})}{\partial z^{(2)}}$
(error term of the hidden layer)

Learning Is An Optimization Problem

- Update the **weights** and **biases** to decrease **loss function**



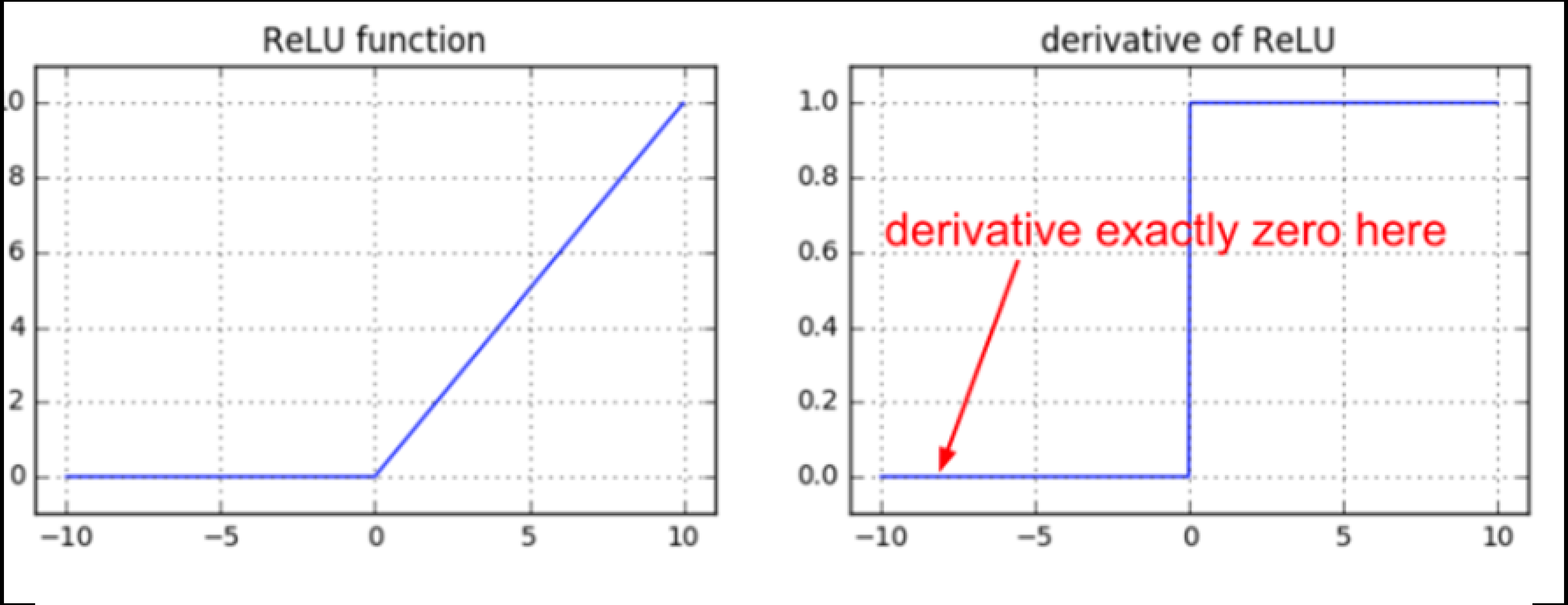
Optimization Is Hard: Vanishing Gradients



$$\frac{d\sigma(x)}{dx} = (1 - \sigma(x))\,\sigma(x)$$

Partial derivatives are small = Learning is slow

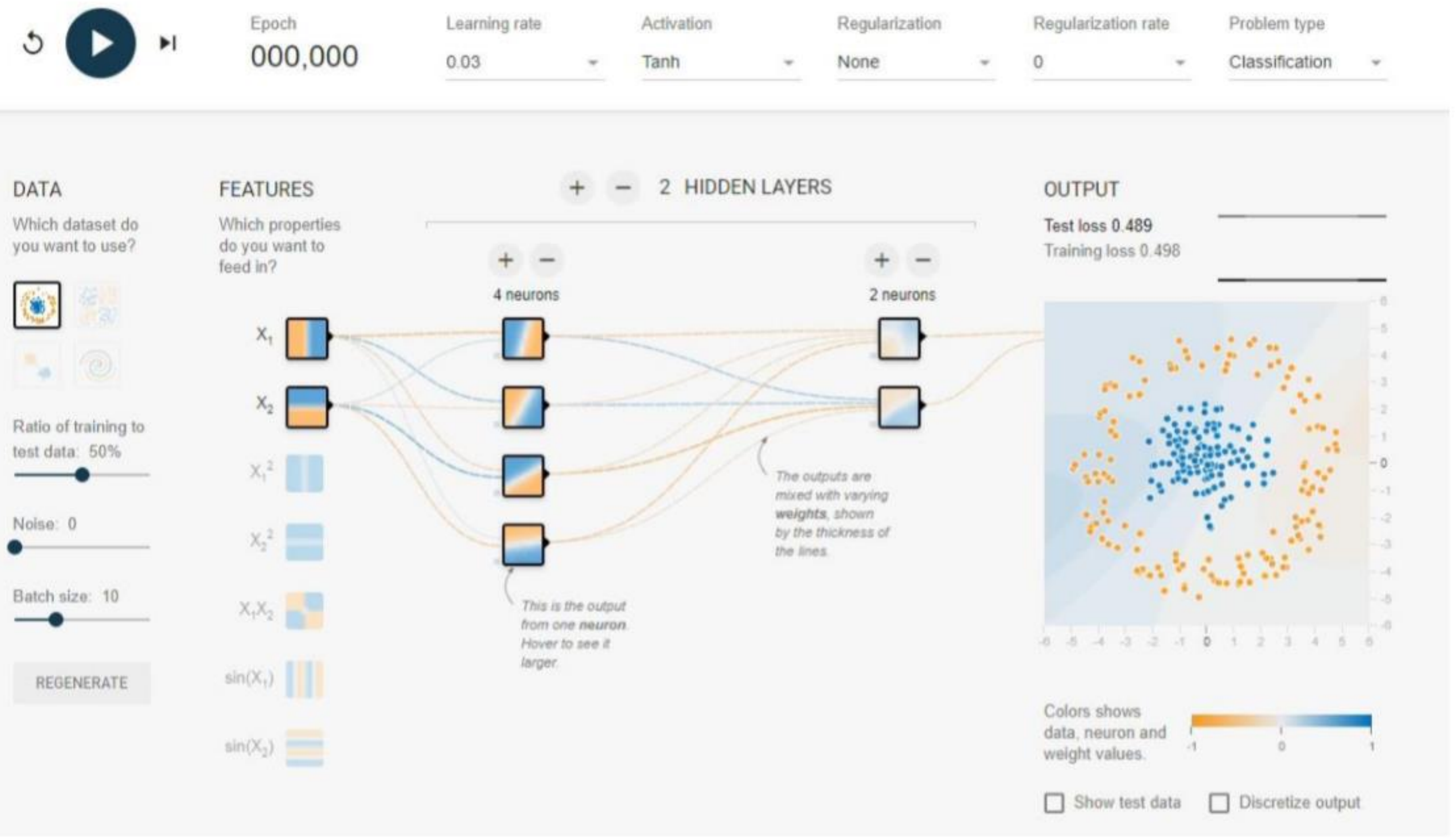
Optimization Is Hard: Dying Relus



- If a neuron is initialized poorly, it might not fire for entire training dataset.
- Large parts of your network could be dead ReLUs!

Neural Network Playground

http://playground.tensorflow.org



Epoch 000,000

Learning rate 0.03

Activation Tanh

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

REGENERATE

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

2 neurons

*This is the output from one **neuron**. Hover to see it larger.*

*The outputs are mixed with varying **weights**, shown by the thickness of the lines.*

OUTPUT

Test loss 0.489

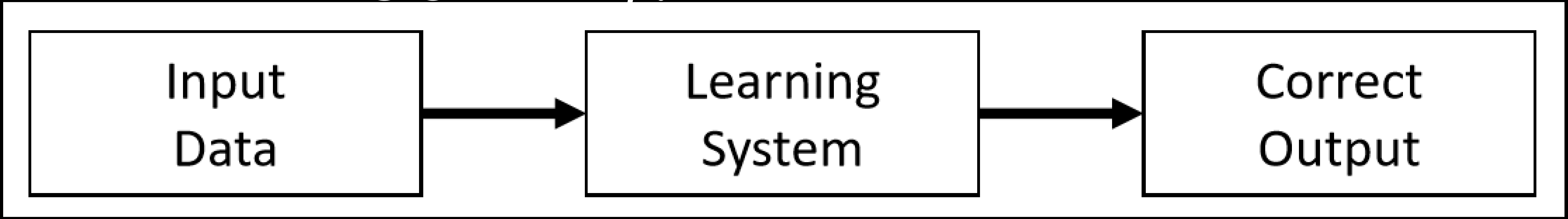
Training loss 0.498

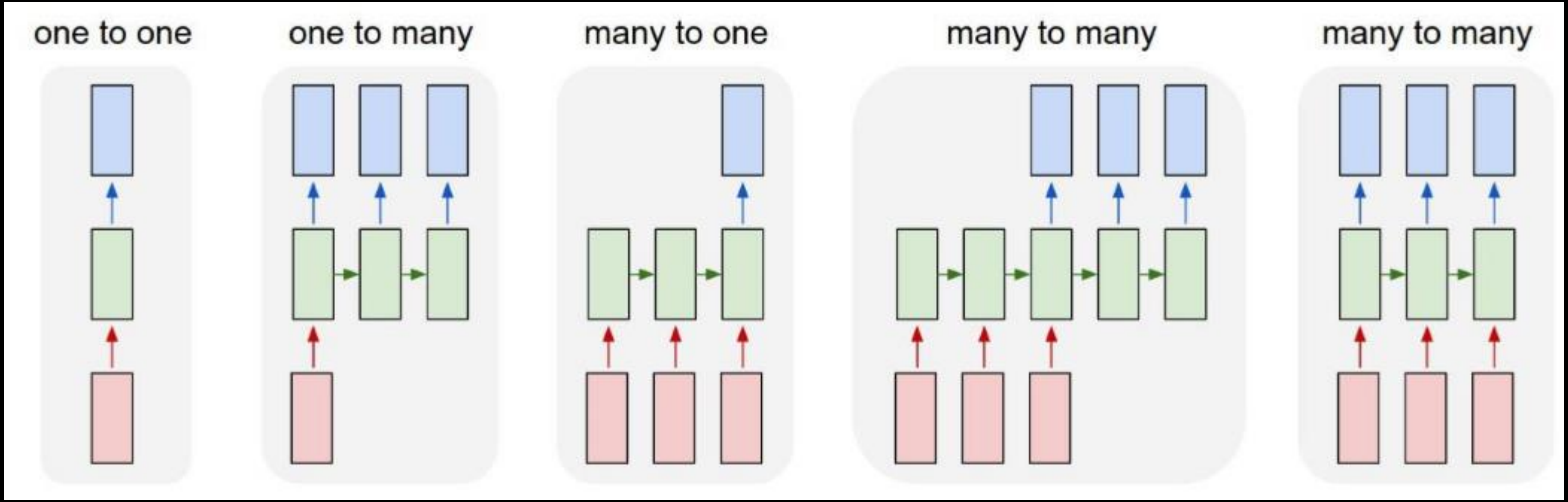
Colors shows data, neuron and weight values.

☐ Show test data

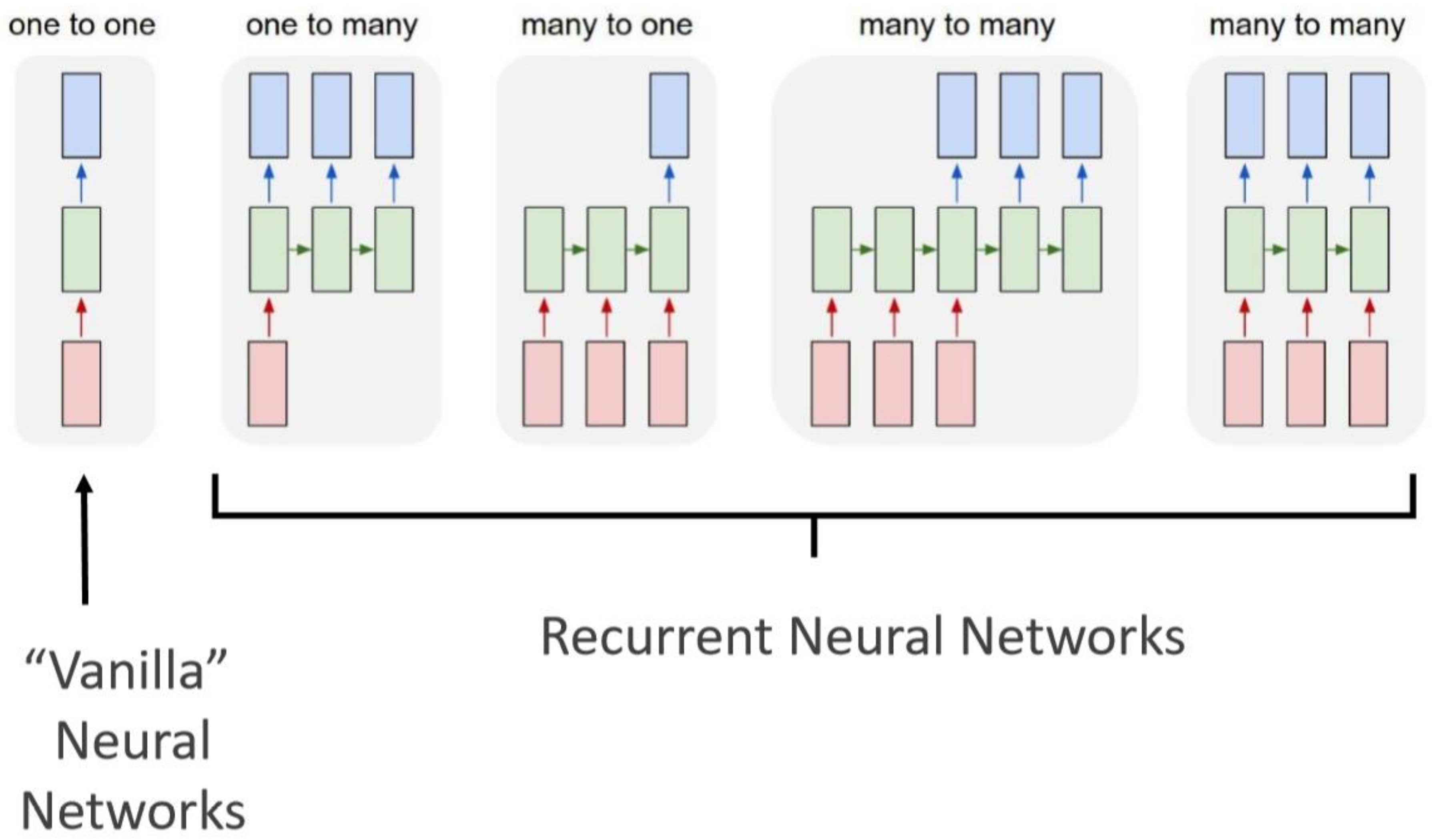
☐ Discretize output

What Can We Do With Deep Learning?

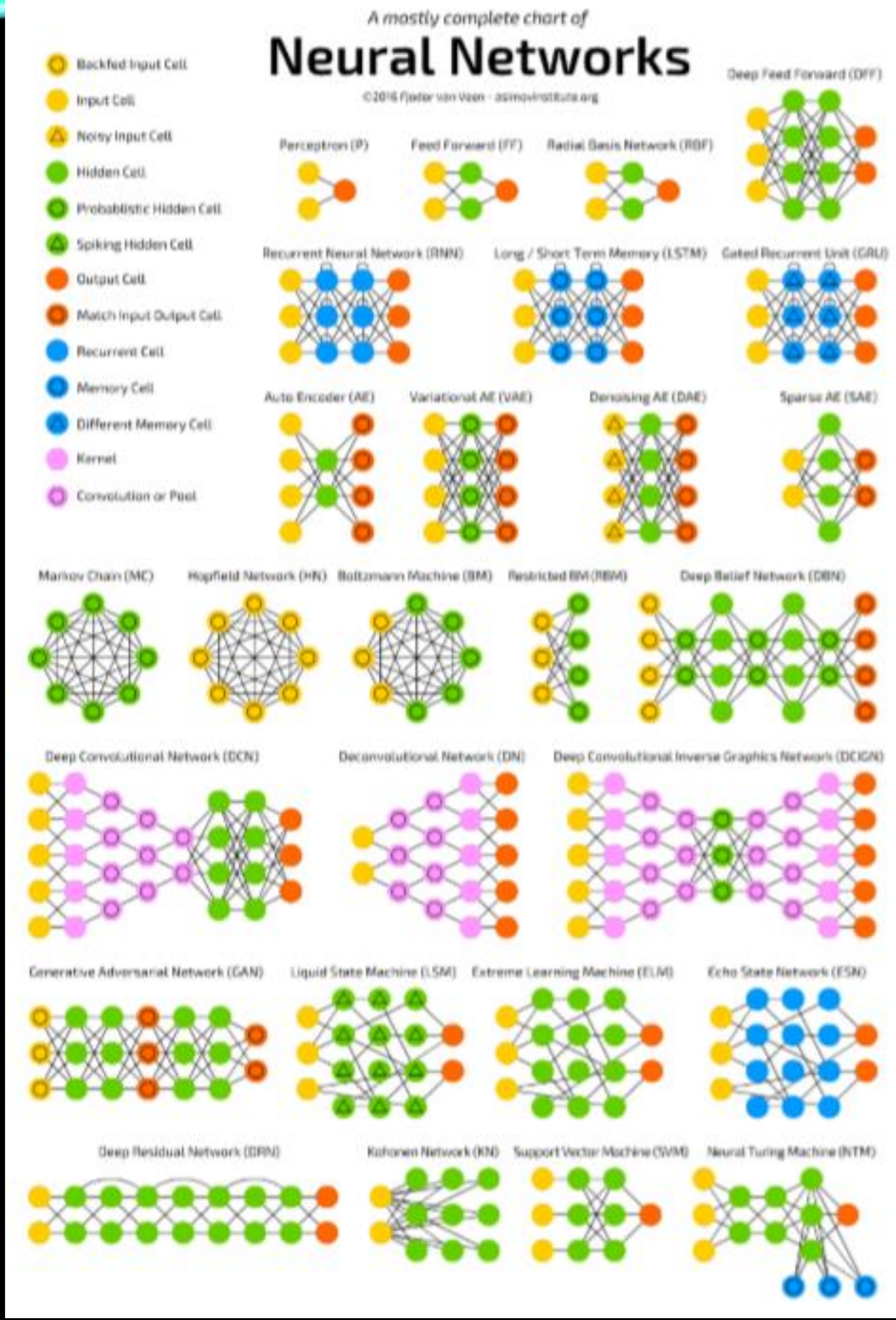




Flavors of Neural Networks

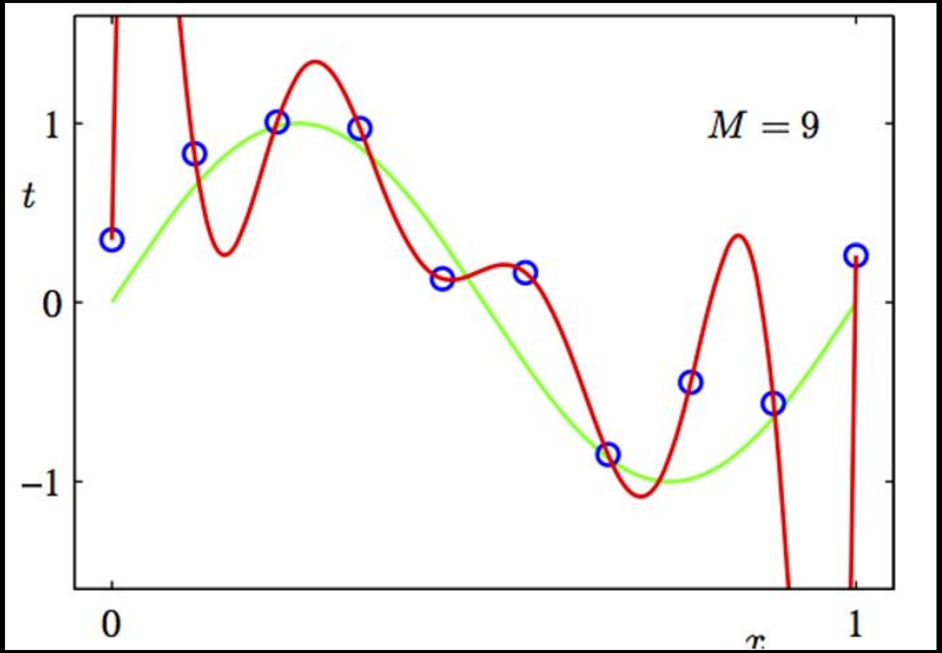


- Basic terms:
  - **Deep Learning ≈ Neural Networks**
  - **Deep Learning** is a subset of **Machine Learning**
- Terms for neural networks:
  - **MLP:** Multilayer Perceptron
  - **DNN:** Deep neural networks
  - **RNN:** Recurrent neural networks
    - **LSTM:** Long Short-Term Memory
  - **CNN:** Convolutional neural networks
  - **DBN:** Deep Belief Networks
- Neural network operations:
  - Convolution
  - Pooling
  - Activation function
  - Backpropagation


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

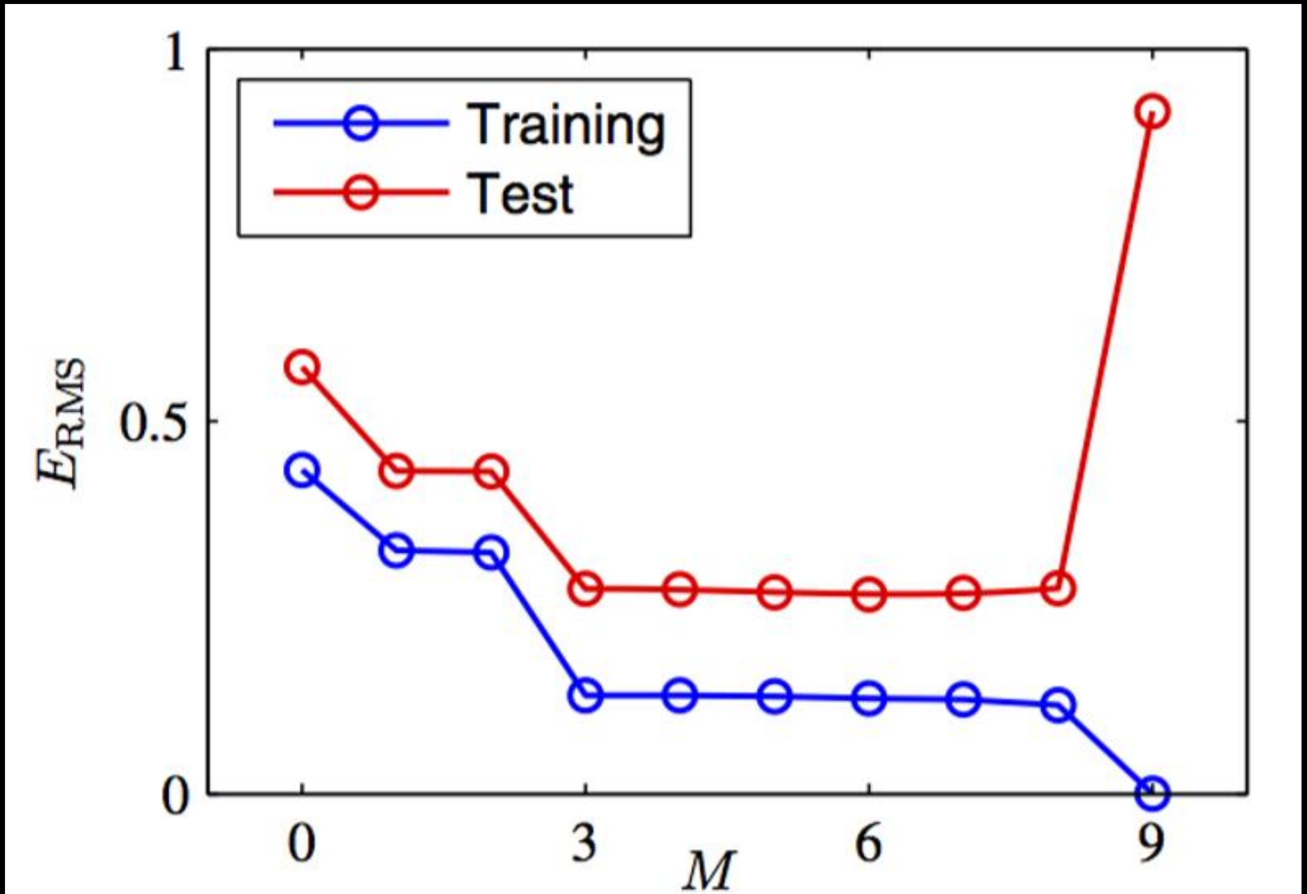
Overfitting and Regularization

- Help the network generalize to data it hasn't seen.
- Big problem for small datasets.
- Overfitting example (a sine curve vs 9-degree polynomial)

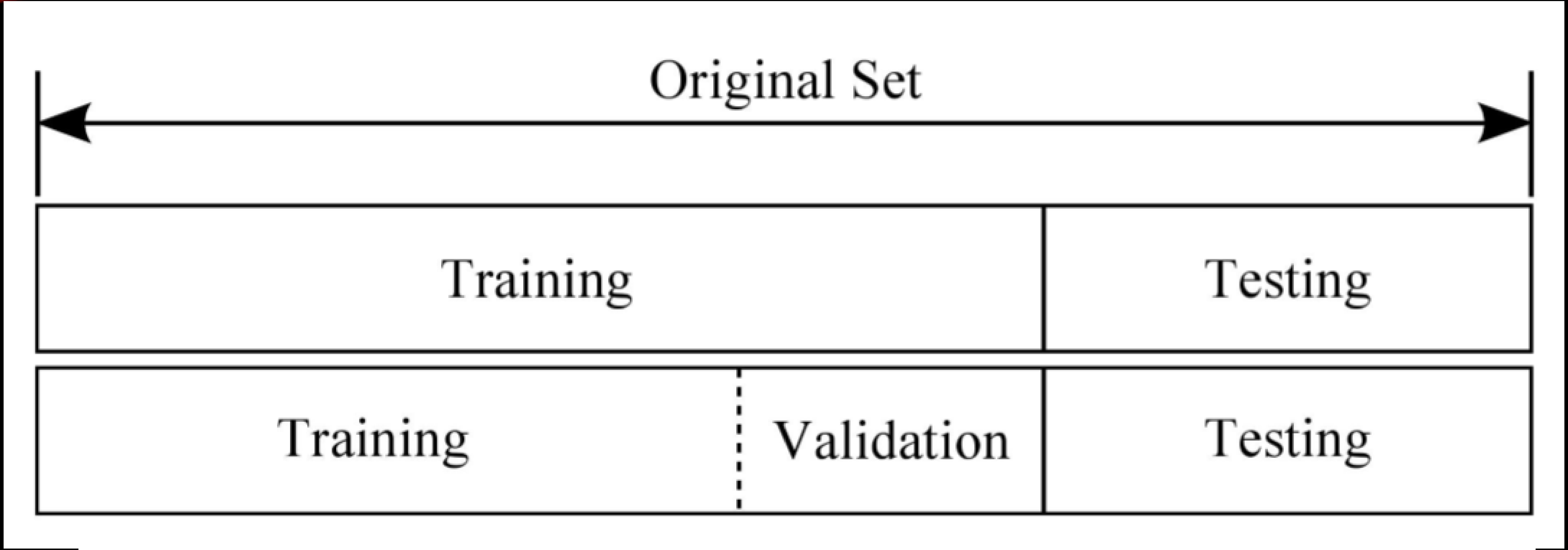


Overfitting And Regularization

- Overfitting: The error decreases in the training set but increases in the test set.



Regularization: Early Stoppage



- Create “validation” set (subset of the training set).
  - Validation set is assumed to be a representative of the testing set.
- **Early stoppage:** Stop training (or at least save a checkpoint) when performance on the validation set decreases