COMP/EECE 7/8745 Neural Networks

Topics:

Introduction of neural networks

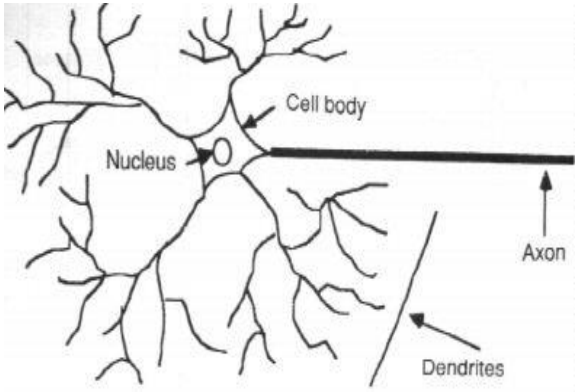
- What is Neural Networks (NNs)?
- NNs with back-propagation (BP) algorithms
- Why Deep NN (DNN)?

Md Zahangir Alom
Department of Computer Science
University of Memphis, TN

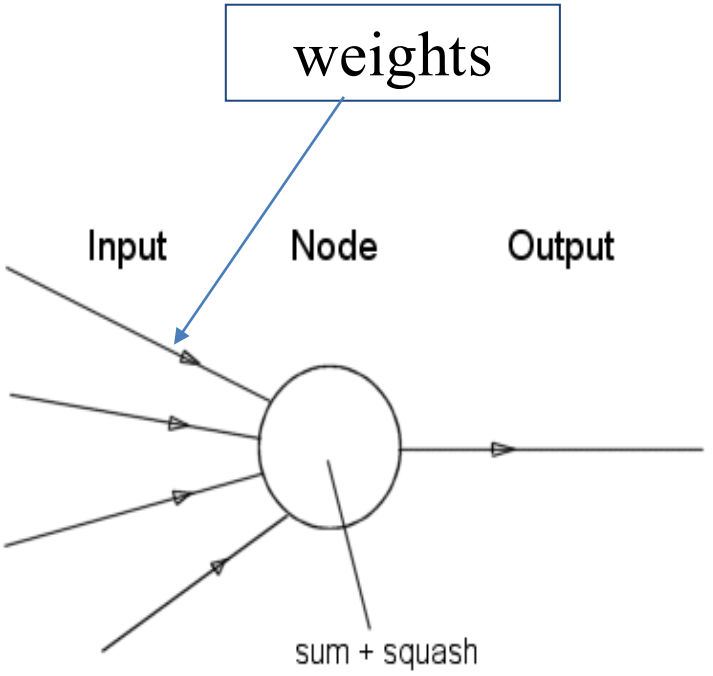
Artificial Neural Networks (ANNs) – The basics

ANNs incorporate the **two fundamental components** of biological neural nets:

1. Neurones (nodes)
2. Synapses (weights)



Biological neurons



Artificial neurons

Axon : A long, thin projection of a neuron

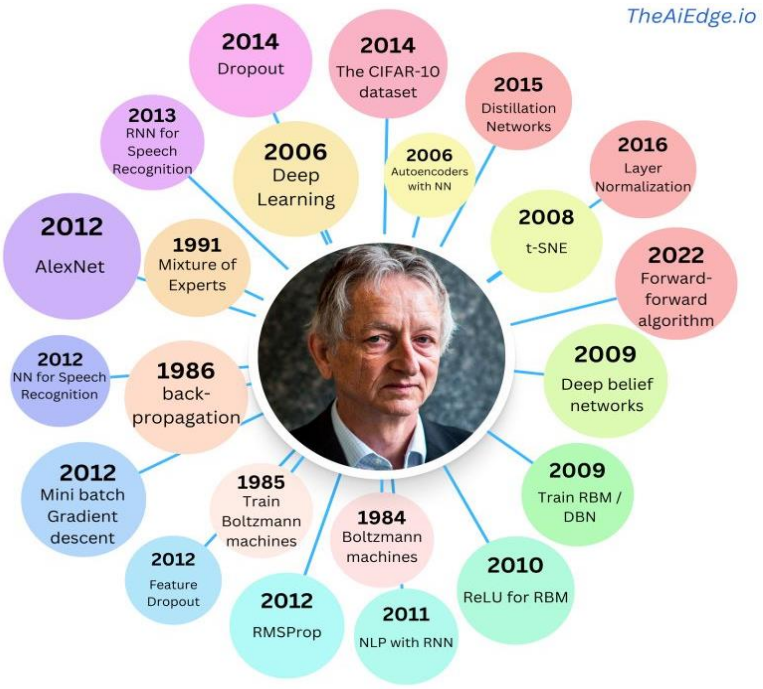
Synapses: A junction between two neurons

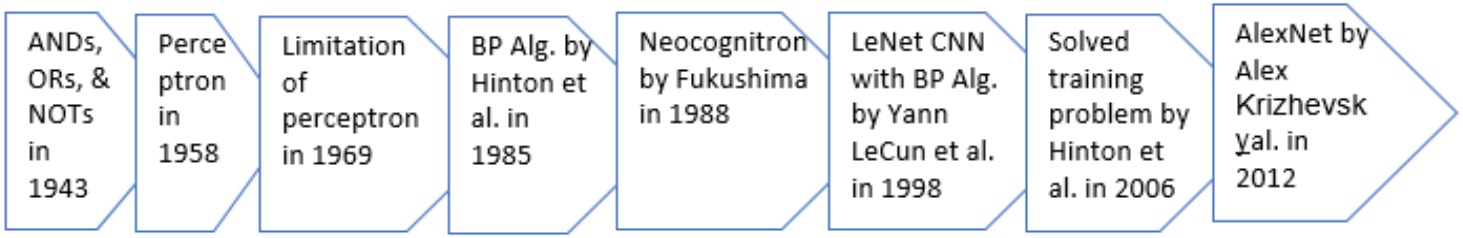
What is Neural Networks?

- **Neural Network (NN)** is a machine learning approach inspired by the structure of the human brain, consisting of interconnected artificial neurons.
- Neuron in artificial NNs tend to have **fewer connections than biological neurons**.
- Each neuron computes **a weighted sum of its inputs**, which is then passed through an **activation function** to determine its output.
- Learning occurs through **training examples**, which provide the network with knowledge about the task it needs to perform.

Alom, Md Zahangir, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul AS Awwal, and Vijayan K. Asari. "A state-of-the-art survey on deep learning theory and architectures." *Electronics* 8, no. 3 (2019): 292.

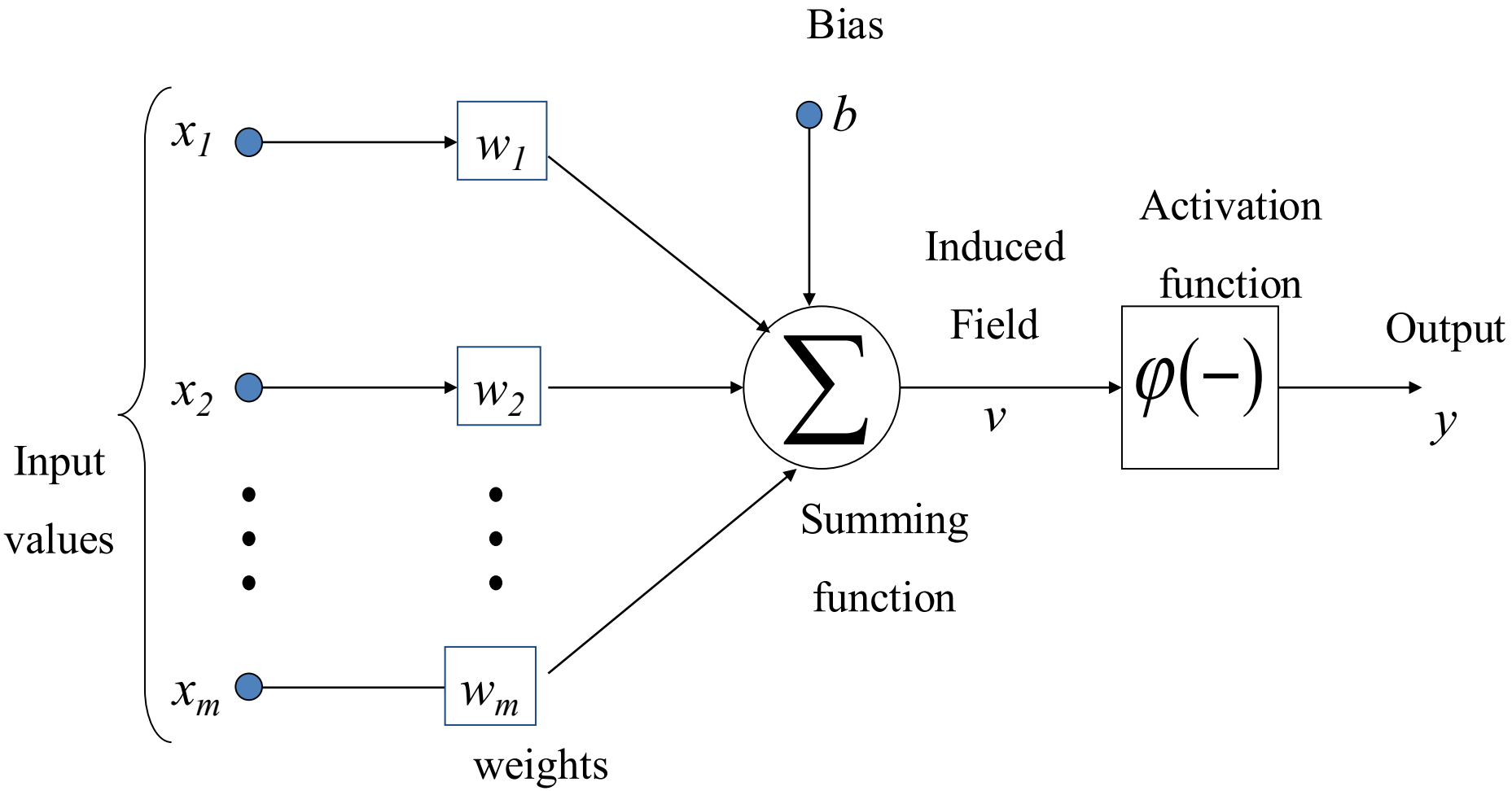
History of NNs Revolution





History of Neural Networks to Deep Learning

The Neuron Diagram



Mathematical model of a neuron

- The neuron is the basic information processing unit of a NN.
- It consists of:
  1. A set of links, describing the neuron inputs, with weights $W_1, W_2, \ldots, W_m$
  2. An adder function (linear combiner) for computing the weighted sum of the inputs: (real numbers)

$$v = \sum_{j=1}^{m} W_j \, x_j$$

  3. Activation function φ for limiting the amplitude of the neuron output. Here 'b' denotes bias.

$$\mathrm{y} = \varphi(\mathrm{u} + b)$$

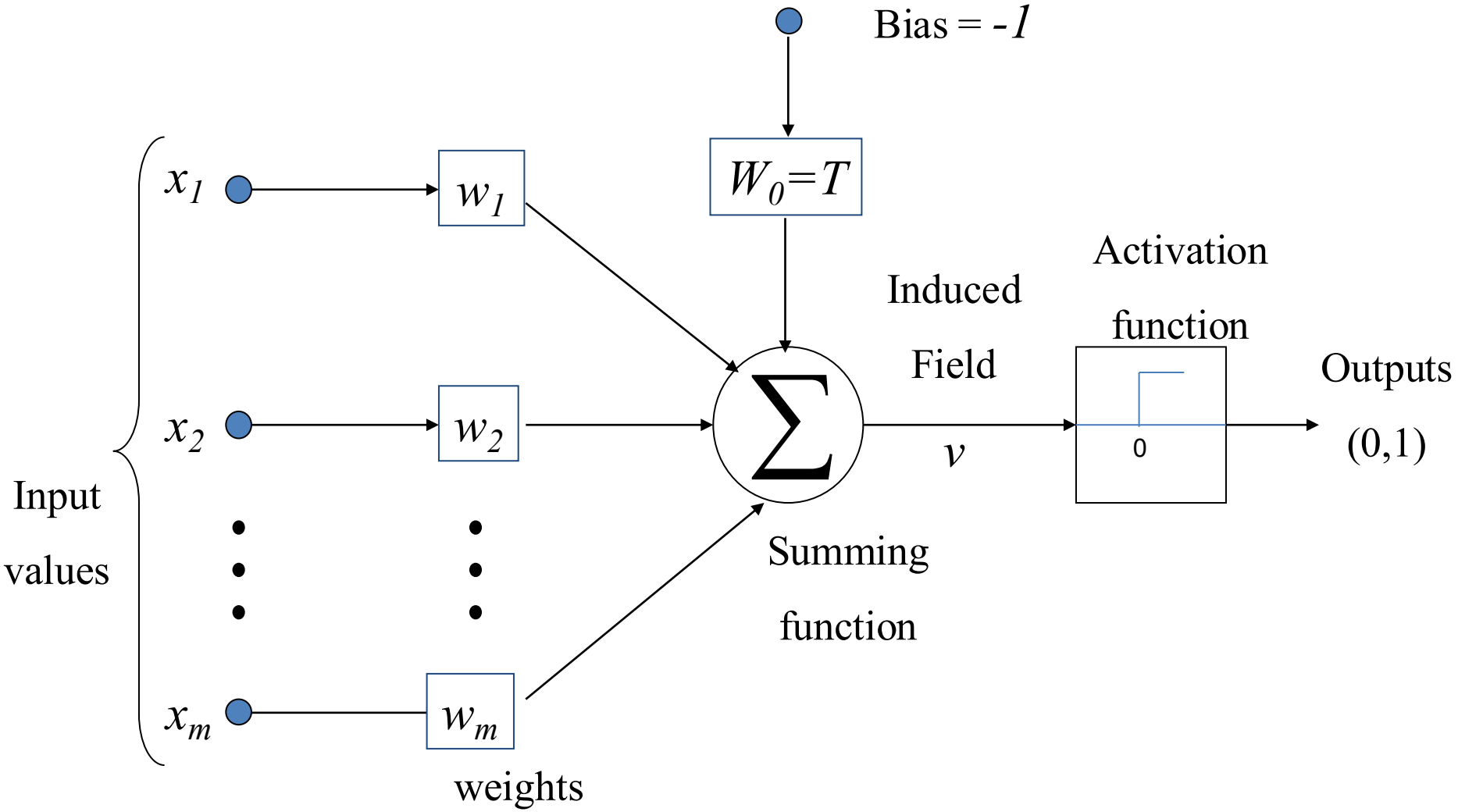
Neuron Models

- The choice of activation function φ determines the neuron model.

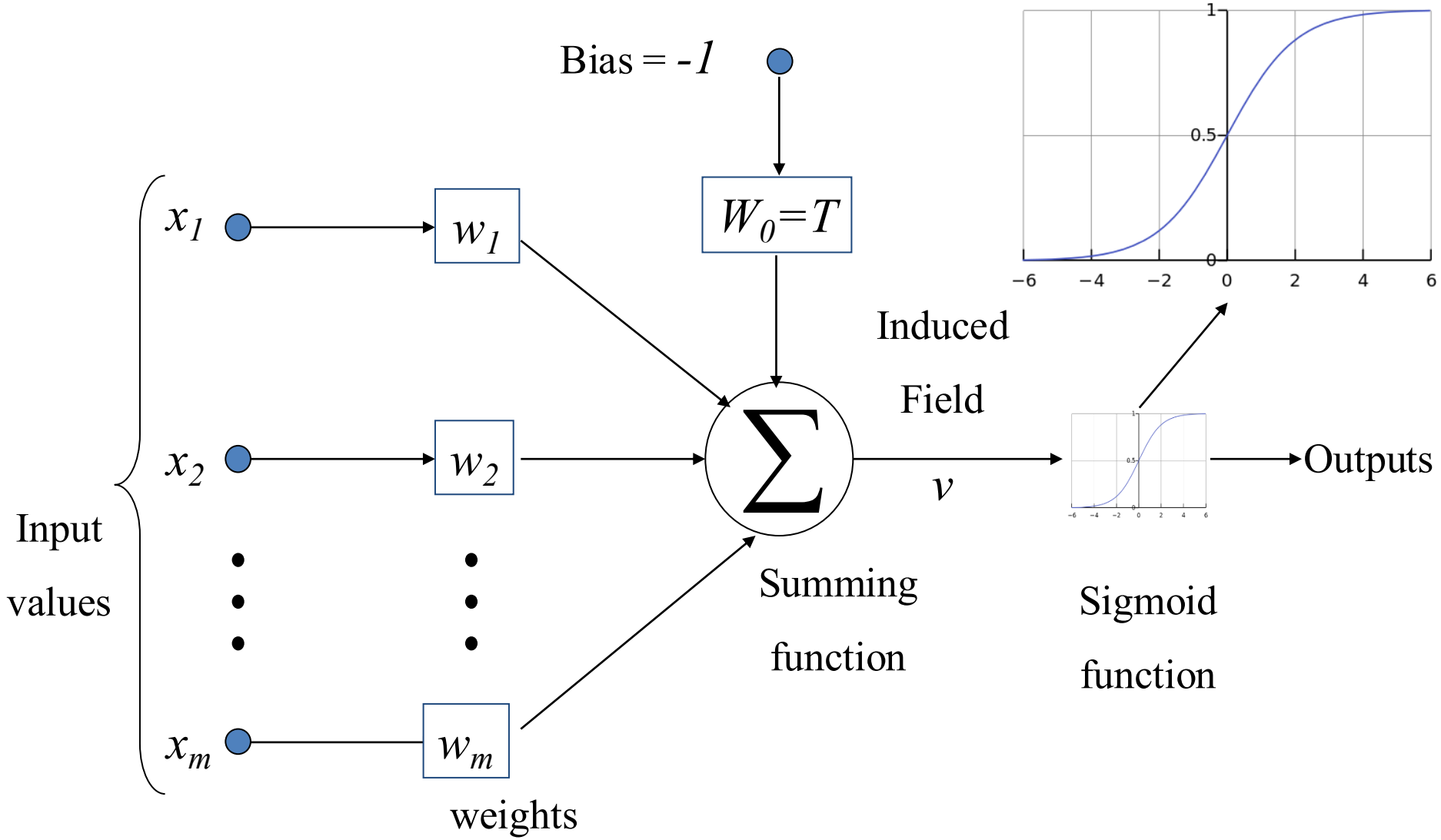
Examples:

- Step function: $$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > c \end{cases}$$
- Ramp function: $$\varphi(v) = \begin{cases} a & \text{if } v < c \\ b & \text{if } v > d \\ a + ((v-c)(b-a)/(d-c)) & \text{otherwise} \end{cases}$$
- Sigmoid function with z, x, y parameters $$\varphi(v) = z + \frac{1}{1 + \exp(-xv + y)}$$
- Gaussian function: $$\varphi(v) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \left(\frac{v-\mu}{\sigma} \right)^2 \right)$$

The Neuron Diagram with activation



The Neuron Diagram with activation

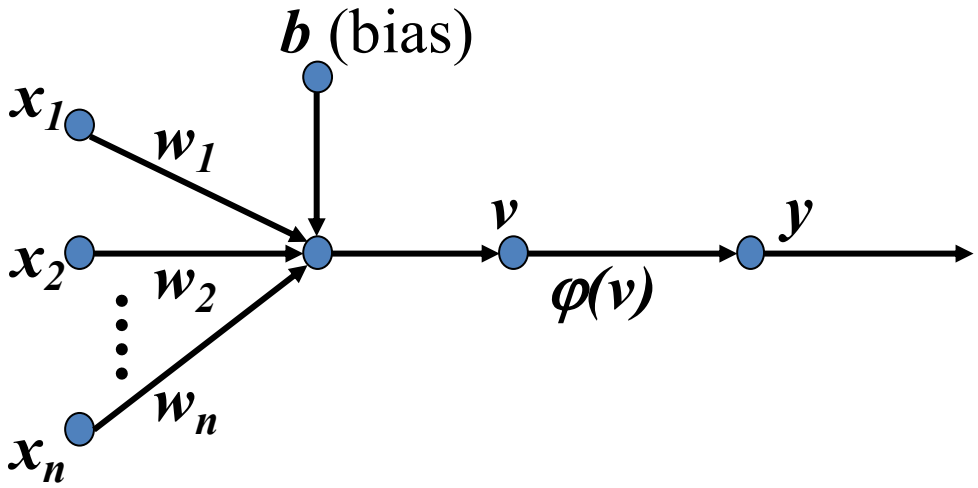


Perceptron: Neuron Model

(Special form of single layer feed forward)

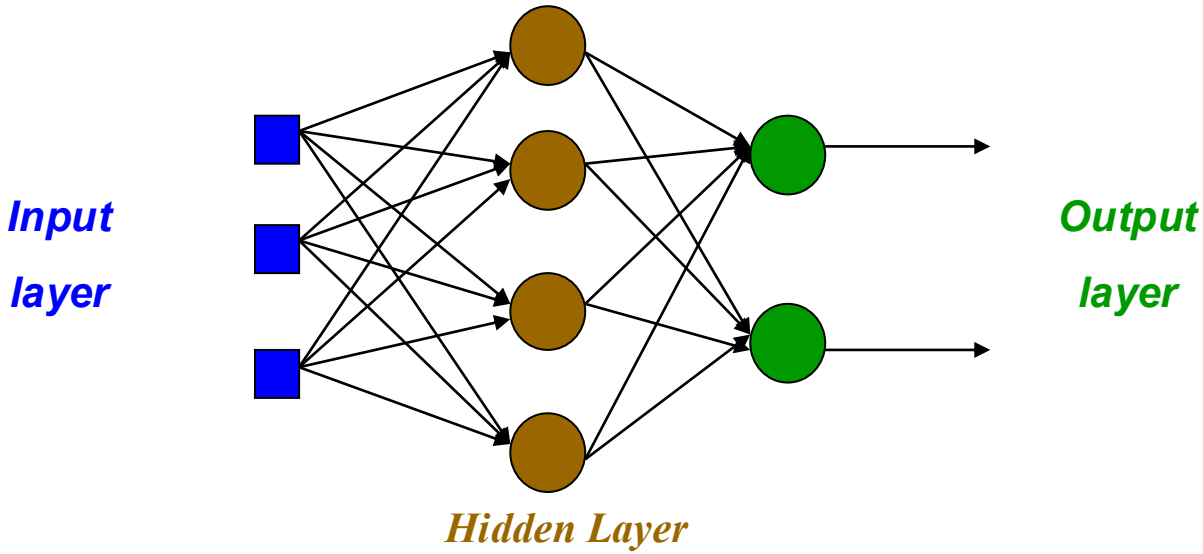
- The perceptron was first proposed by **Rosenblatt (1958)** is a **simple neuron that is used to classify its input into one of two categories.**
- **A perceptron uses a step function** that returns +1 if weighted sum of its input ≥ 0 and -1 otherwise

$$\varphi(v) = \begin{cases} +1 \text{ if } v \geq 0 \\ -1 \text{ if } v < 0 \end{cases}$$



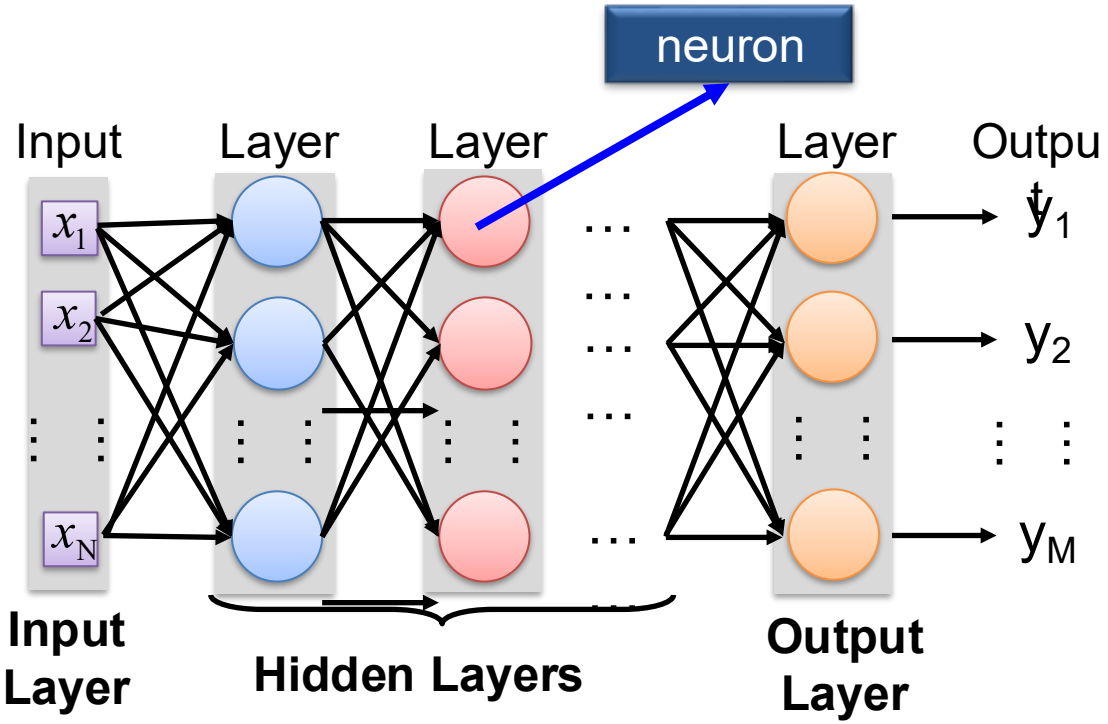
Multi Layer Perceptron (MLP)

- **Feedforward Neural Network (FFNN)** is a more general network architecture consist of hidden layers between input and output layers.
- **Hidden nodes** do not directly receive inputs nor send outputs to the external environment.
- FFNNs overcome the limitation of single-layer NN.
- Can **handle non-linearly separable learning tasks.**



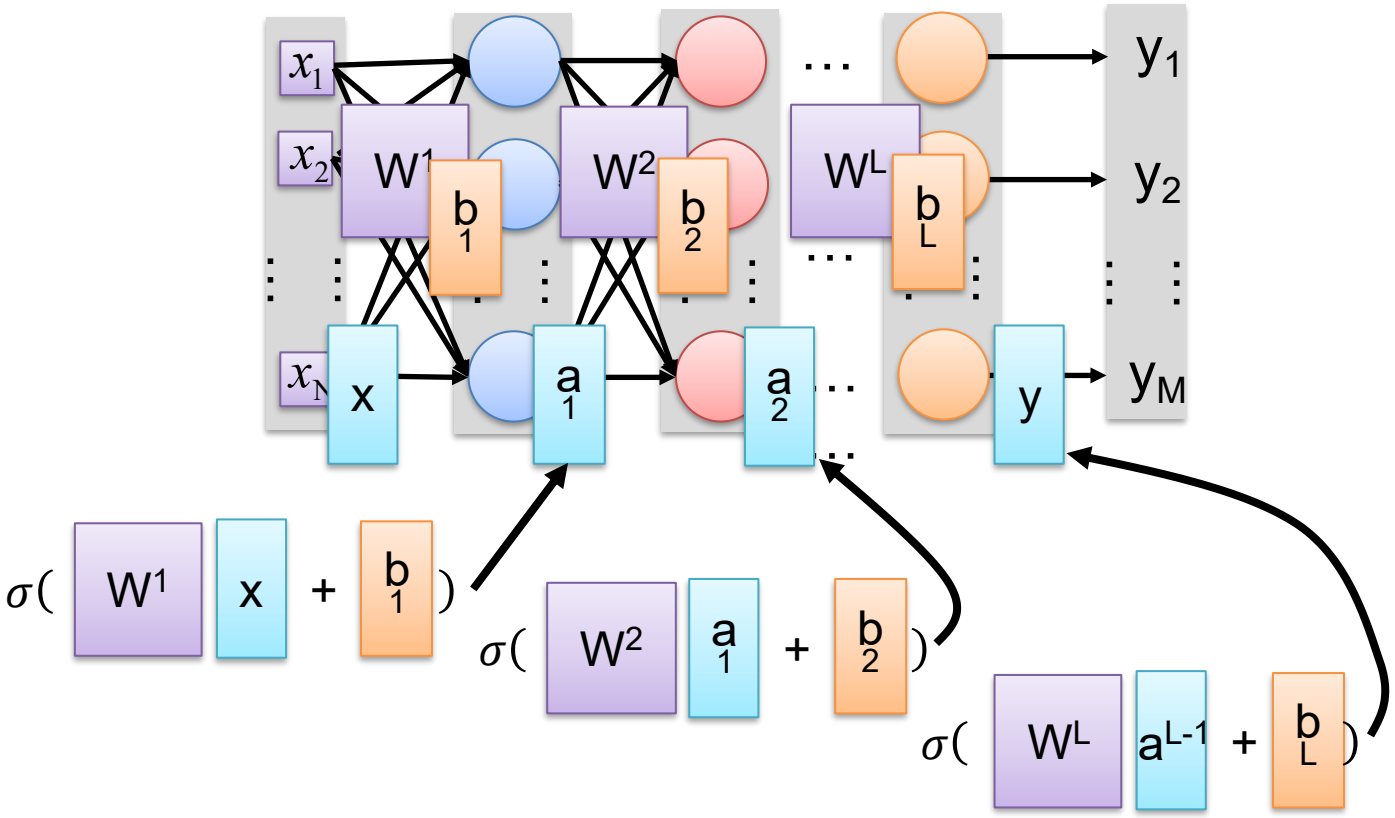
(3-4-2) Network

Deep Neural Networks

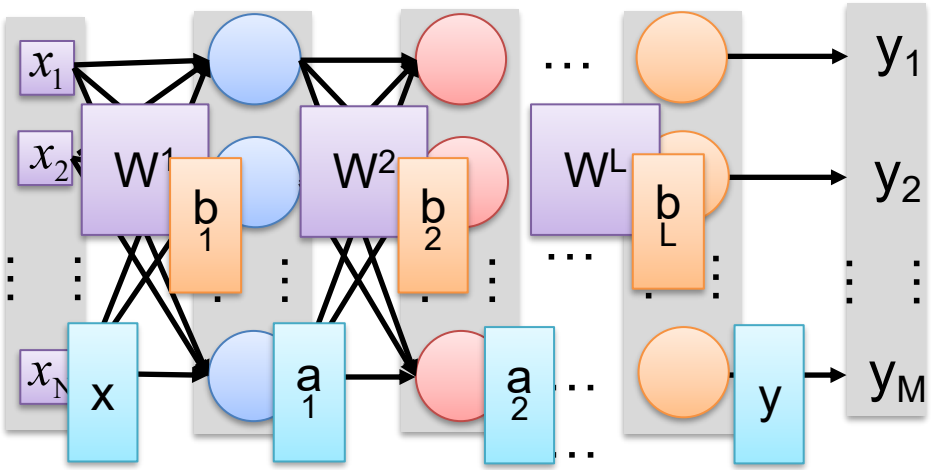


Deep means many hidden layers

Deep Neural Networks



Deep Neural Networks

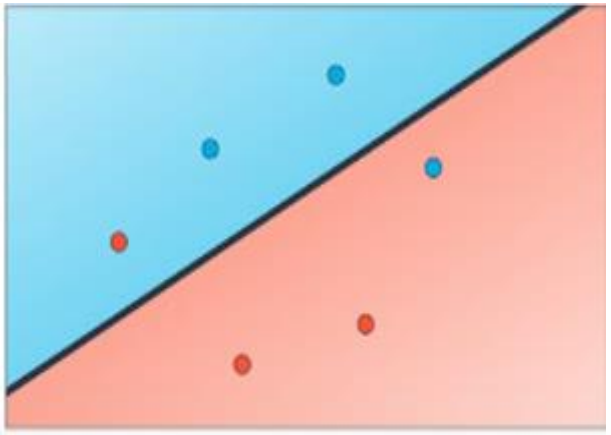


$y = f(x)$

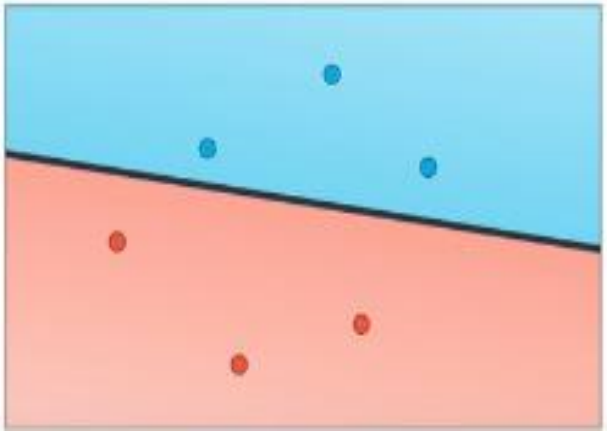
Using parallel computing techniques to speed up matrix operation

$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b_1) + b_2) \cdots + b_L)$

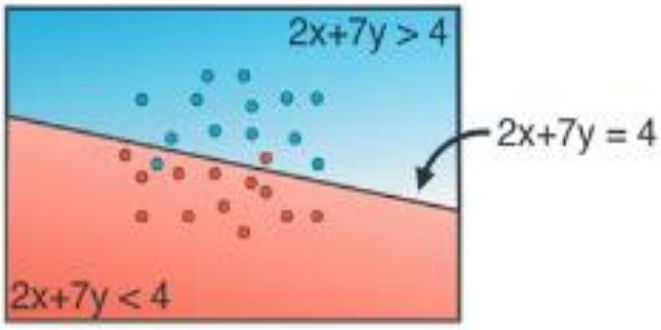
Understanding Neural Networks Through Visualization

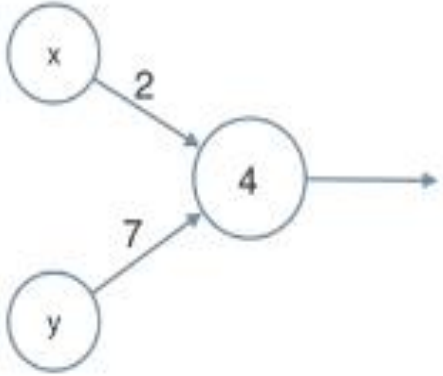


Goal: split data

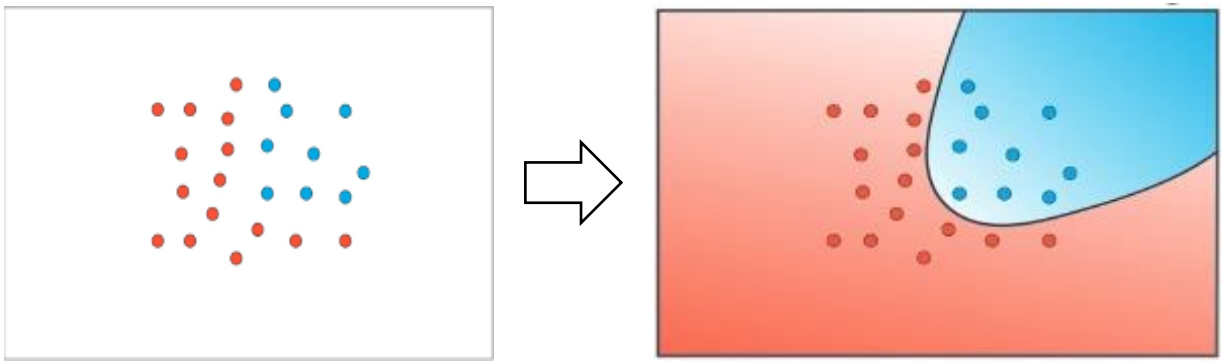


Split data with 0 errors HOT !!!

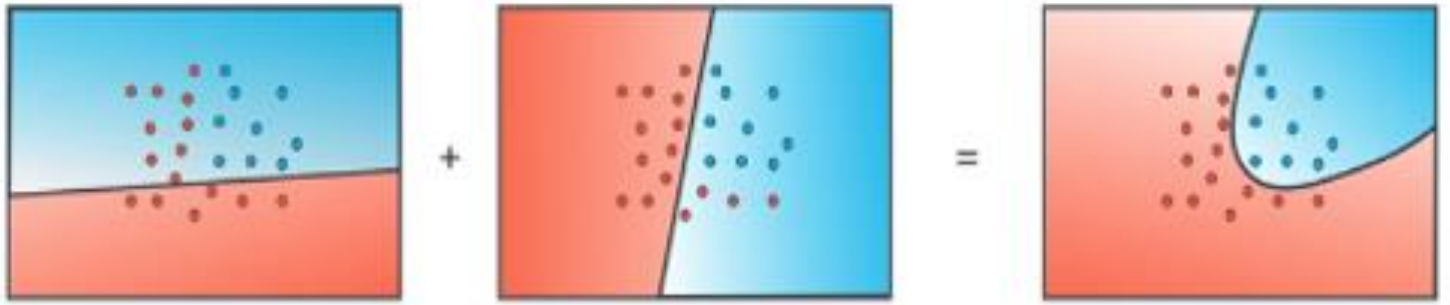




Understanding Neural Networks Through Visualization



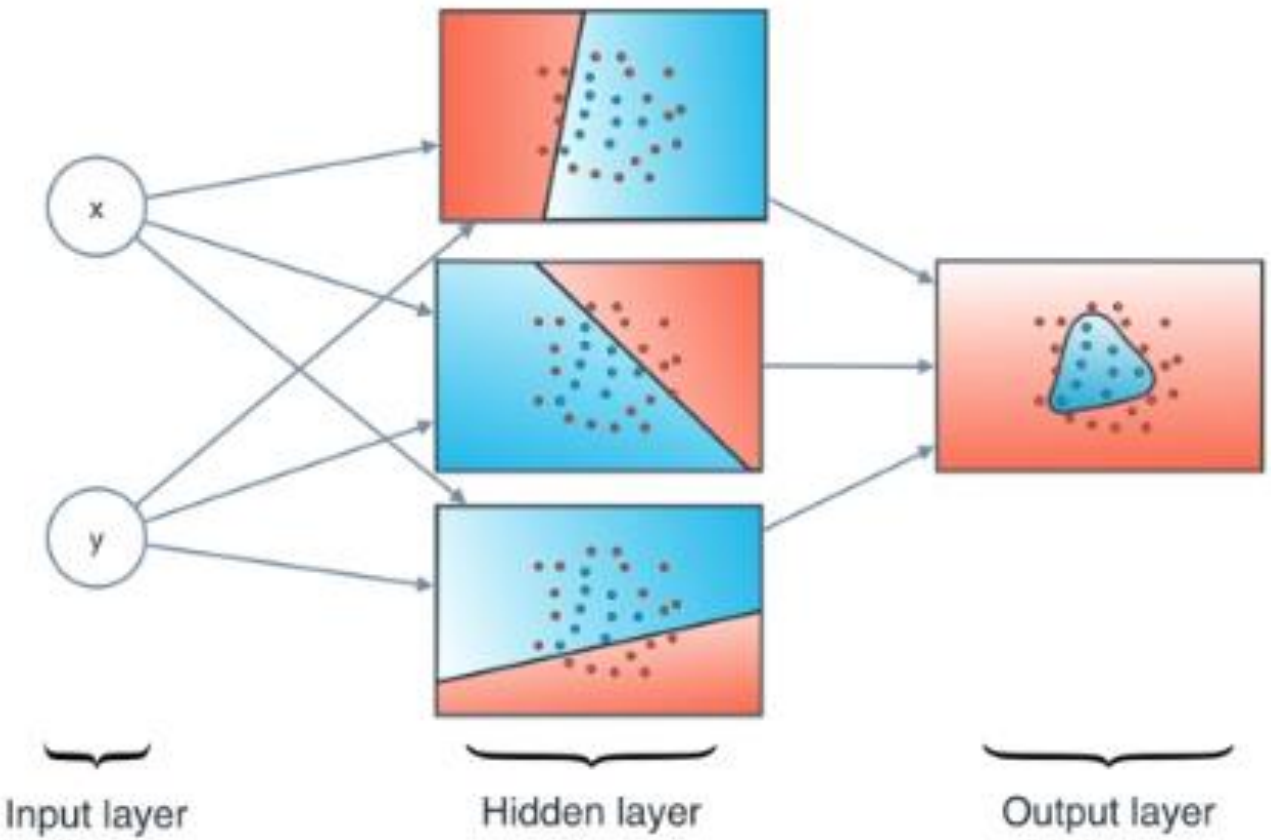
Non-linear regions



Combining regions

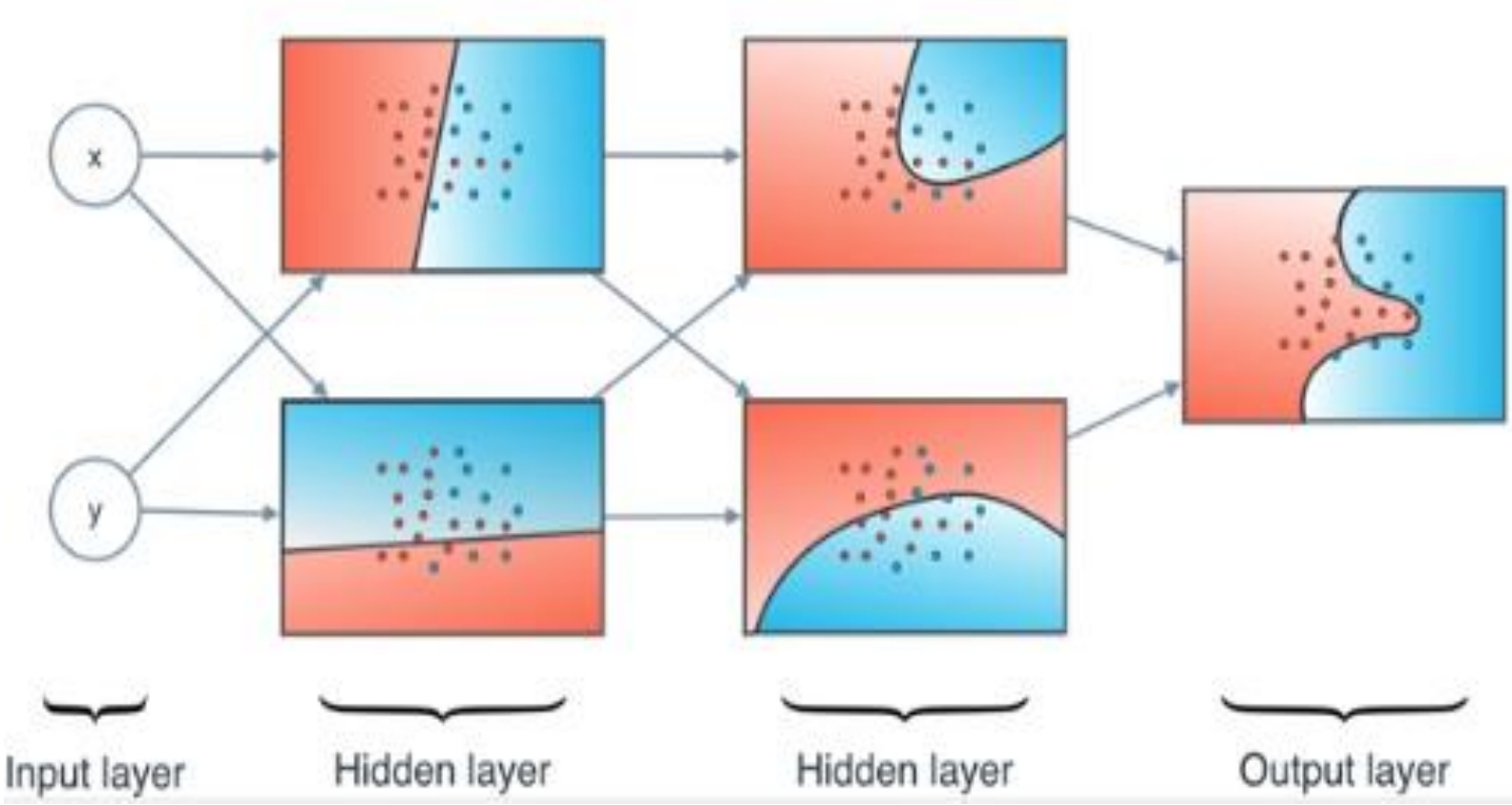
Understanding Neural Networks Through Visualization

- In case of two inputs



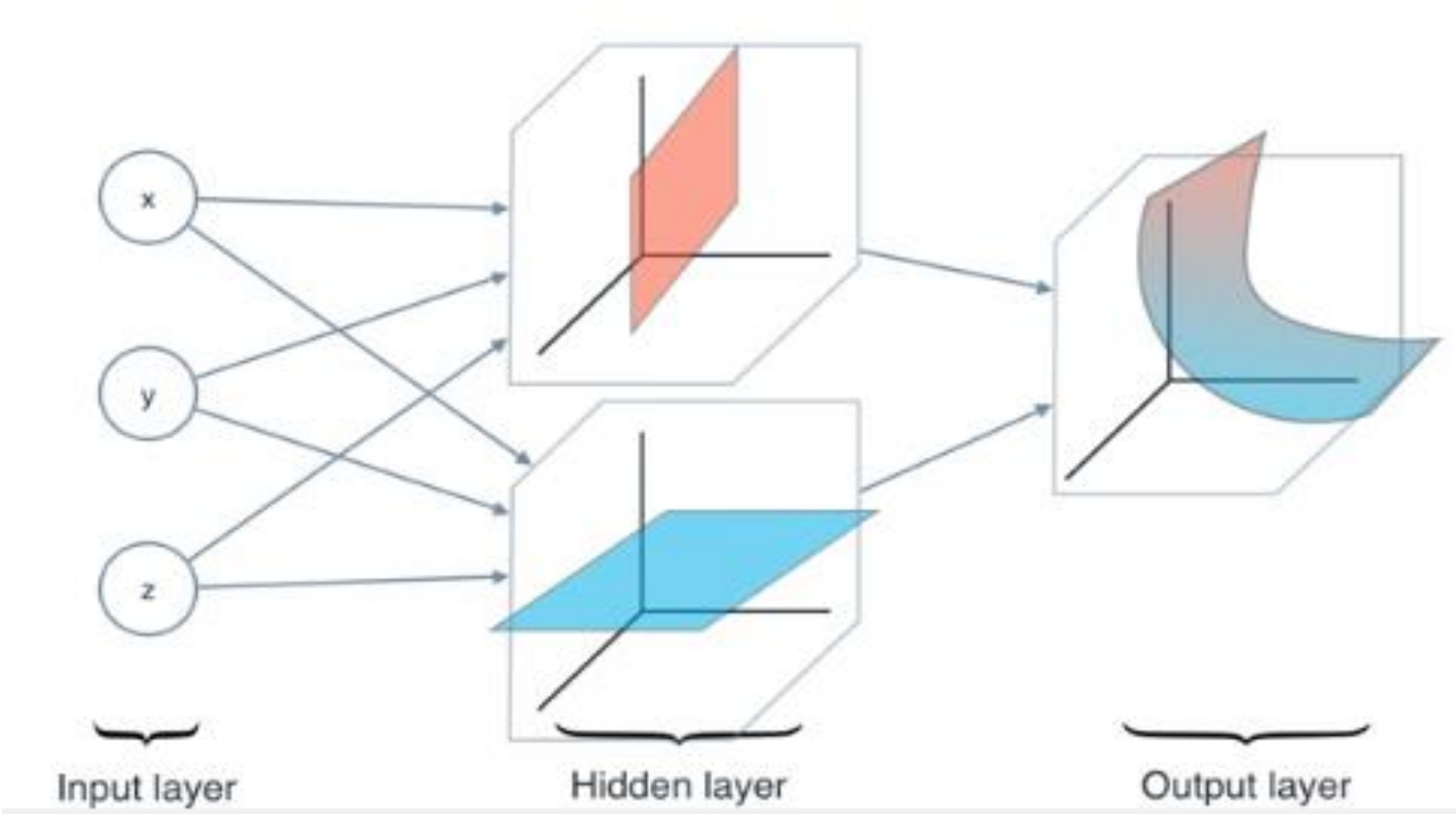
Understanding Neural Networks Through Visualization

- Neural Network with more hidden layers

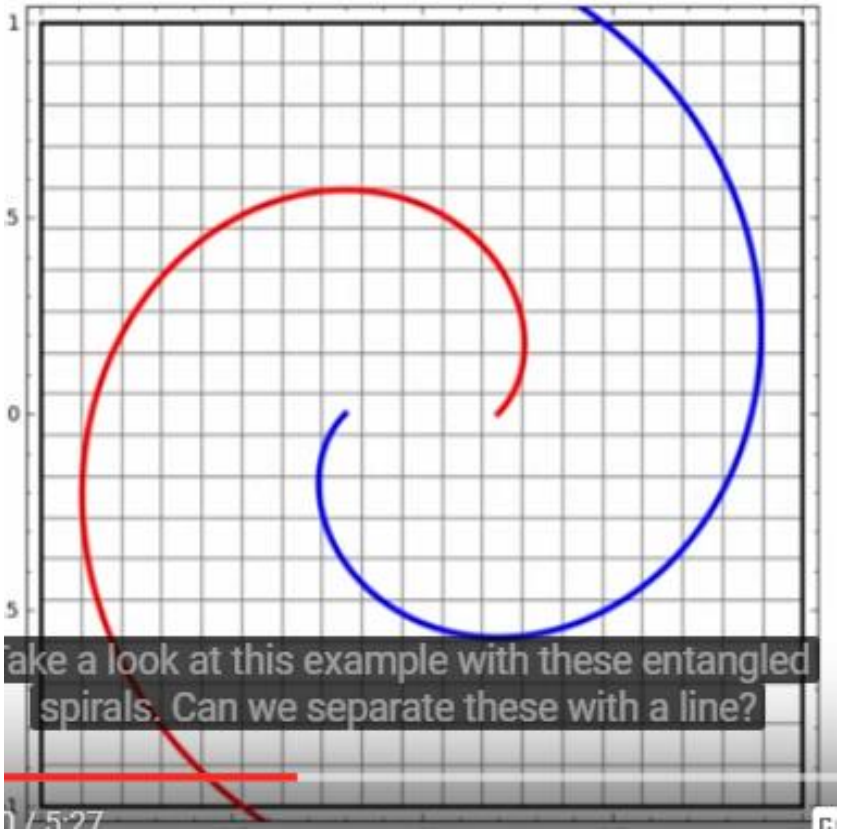


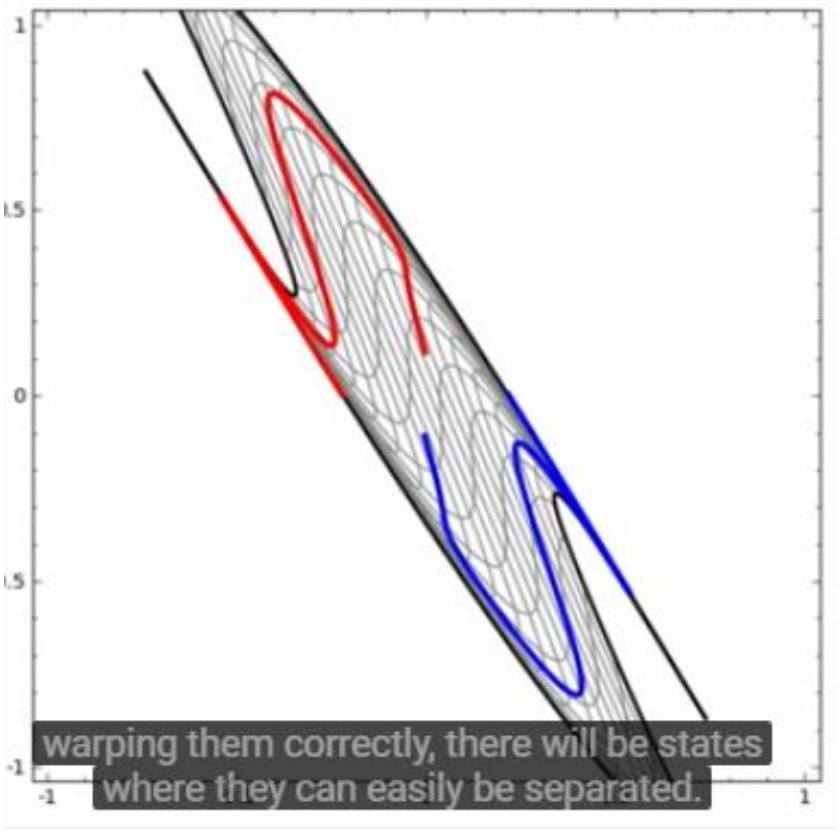
Understanding Neural Networks Through Visualization

- In case of three inputs



Understanding Neural Networks Through Visualization

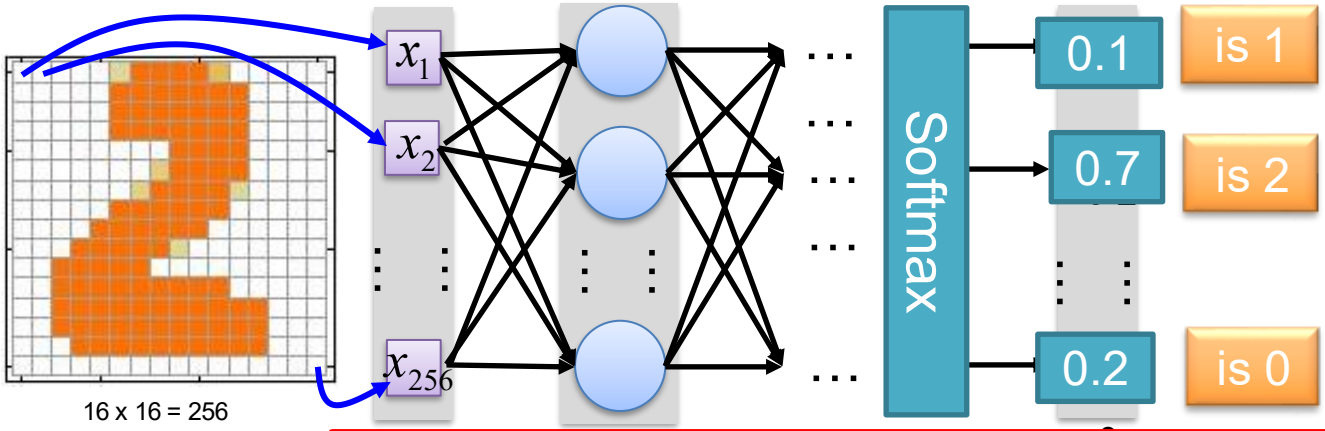




How to set network weights (parameters)

- **Weight** settings determine the behaviour of a network

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



Ink → 1

No ink → 0

Set the network parameters θ such that ……

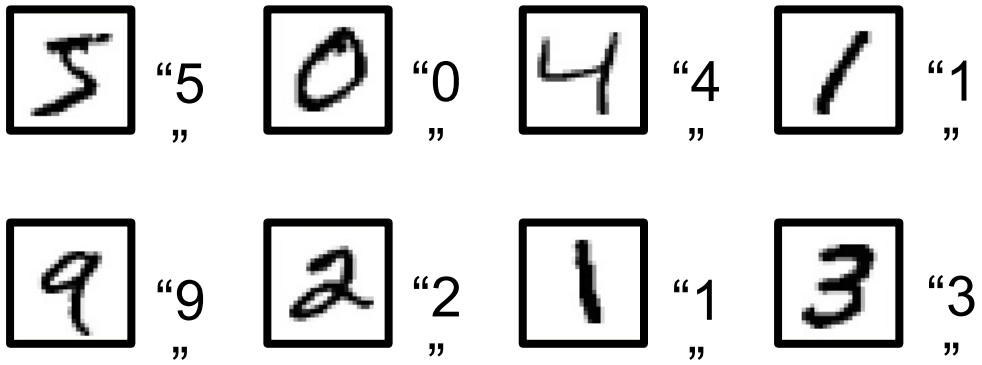
Input:

Input:

How to let the neural network achieve this

Training Data

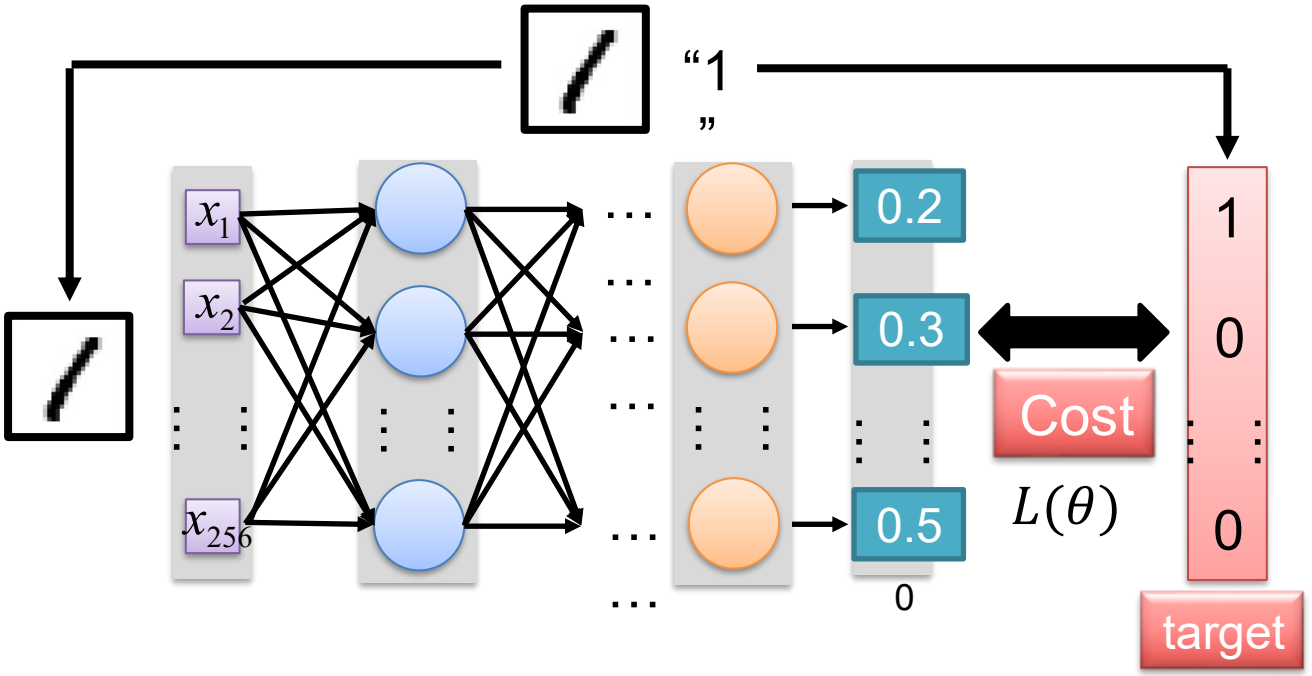
- Preparing training data: images and their labels



Using the training data to find the network parameters.

Training data and cost

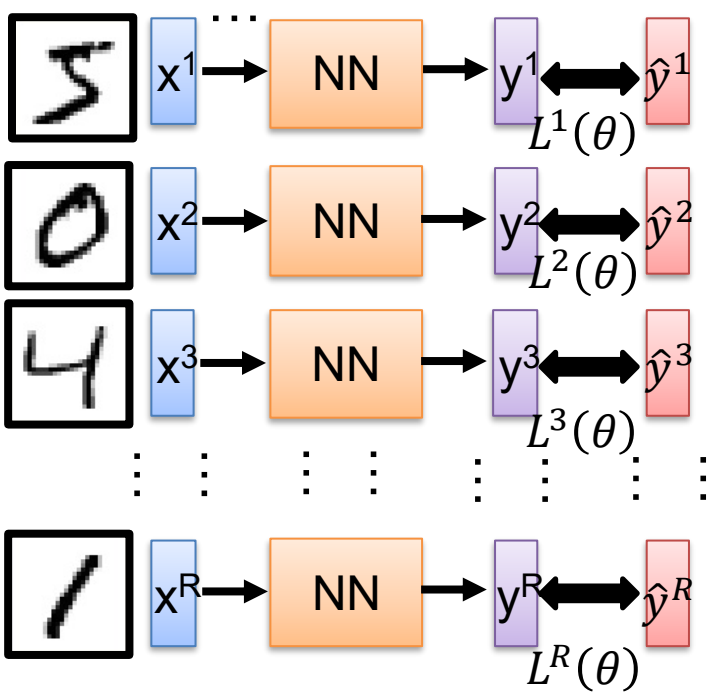
Given a set of network parameters θ, each example has a cost value.



- Cost can be **Euclidean distance** or **cross entropy** of the network output and target

Total Cost

For all training data



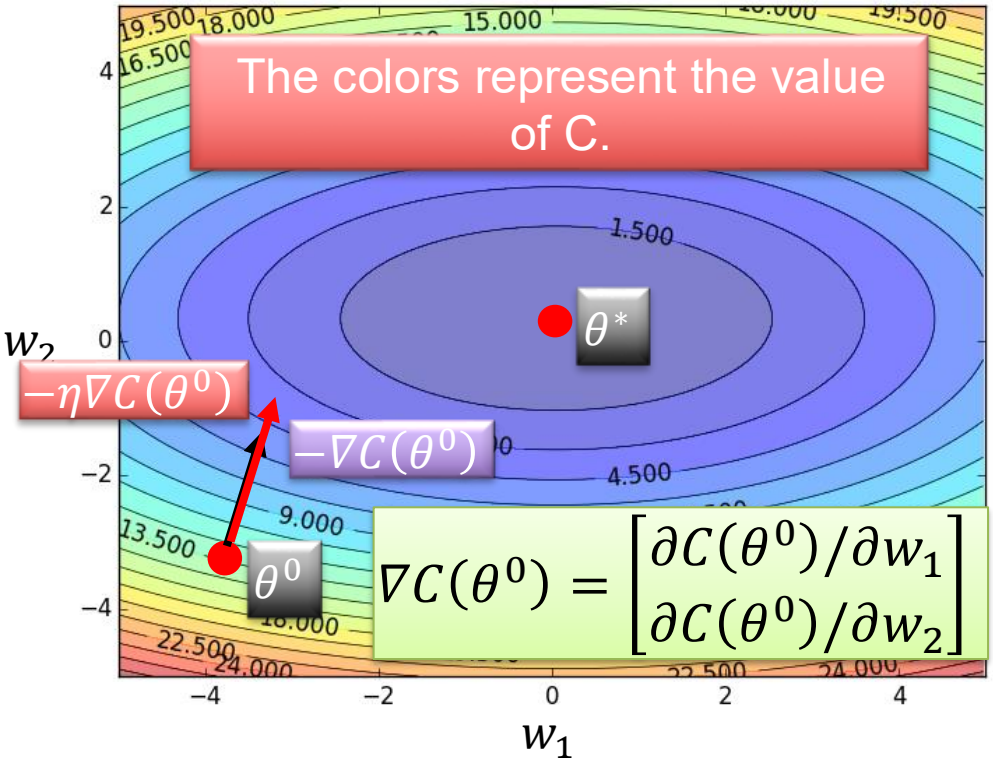
Total Cost:

$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

How bad the network parameters θ is on this task

Find the network parameters θ^* that **minimize this value**

Gradient Descent (GD)



Error Surface

Assume there are only two parameters **w_1 and w_2** in a network.

$\theta = \{w_1, w_2\}$

Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

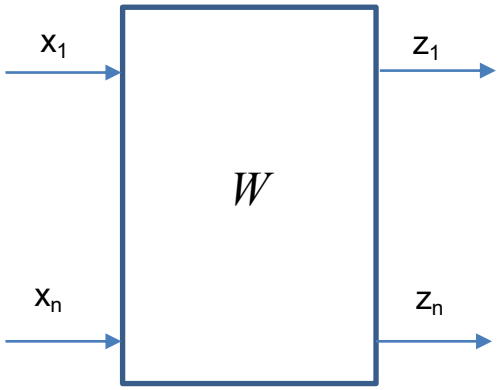
$-\nabla C(\theta^0)$

Times the learning rate η

$-\eta\nabla C(\theta^0)$

Mathematical model of GD

- **Neural network:** input / output transformation



$$z = F(x, W)$$

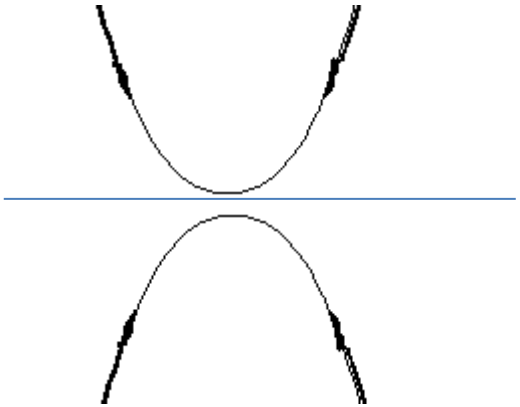
W is the matrix of all weight vectors, and **d is the targets:**

$$d = G(x)$$

Performance function:

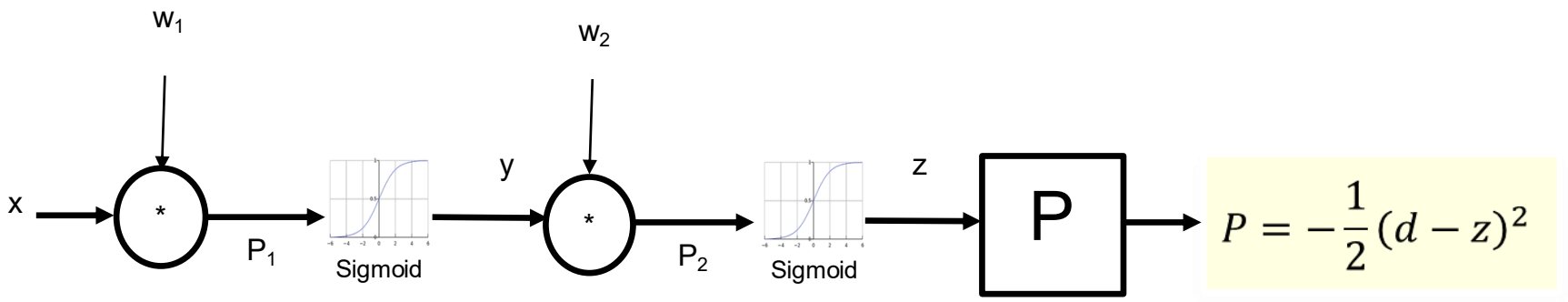
$$P = -\frac{1}{2}\|d - z\|^2$$

Gradient descent

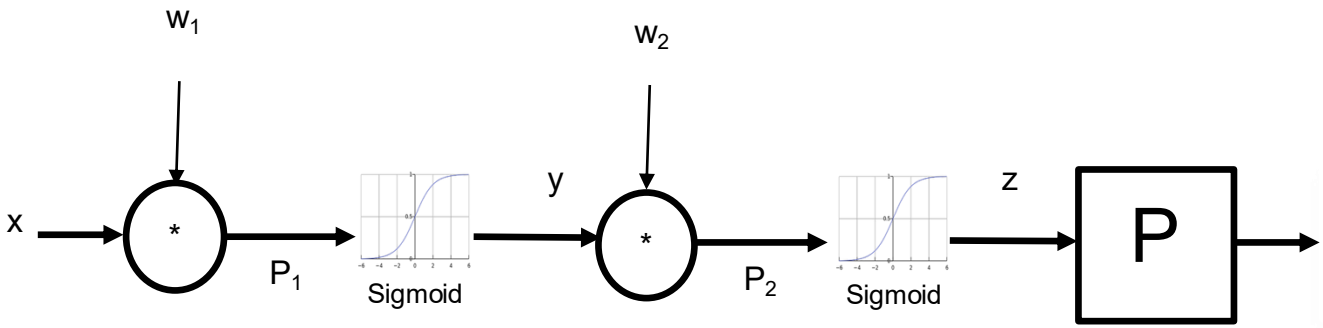


Gradient ascent

Simplest MLP

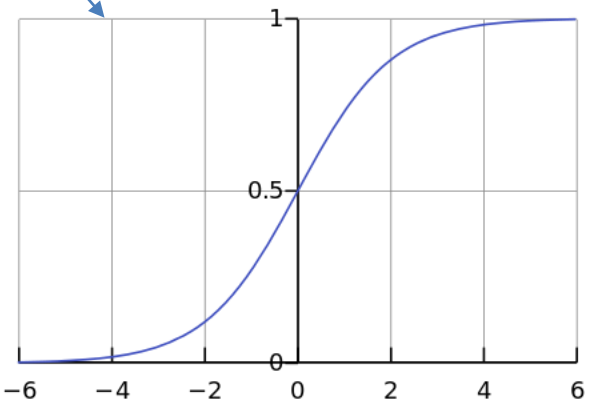


Simplest MLP

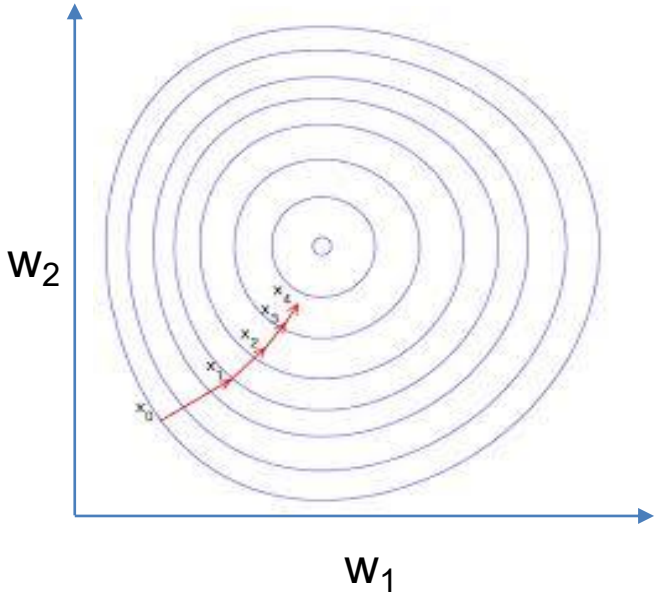


$$P = -\frac{1}{2}(d - z)^2$$

$$\frac{1}{1+e^{-\alpha}}$$

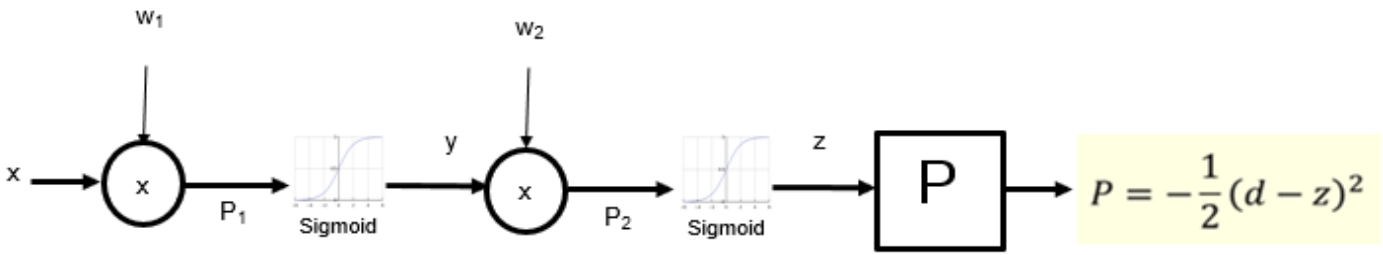


Gradient ascent (Hill Climbing Approach)



$$\Delta w = \eta\left(\frac{\partial P}{\partial x} i + \frac{\partial P}{\partial y} j\right) \quad = \eta\left(\frac{\partial P}{\partial w_1} + \frac{\partial P}{\partial w_2}\right)$$

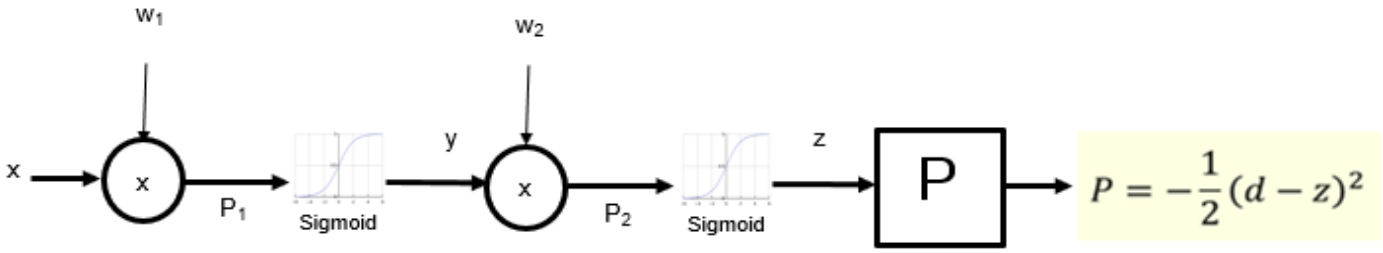
Weight update..



$$\frac{\partial P}{\partial w_2} = \frac{\partial P}{\partial z}\frac{\partial z}{\partial w_2} = (d-z)\frac{\partial z}{\partial w_2}$$

$$= (d-z)\frac{\partial z}{\partial p_2}\frac{\partial p_2}{\partial w_2} = (d-z)\frac{\partial z}{\partial p_2}y$$

Weight update..



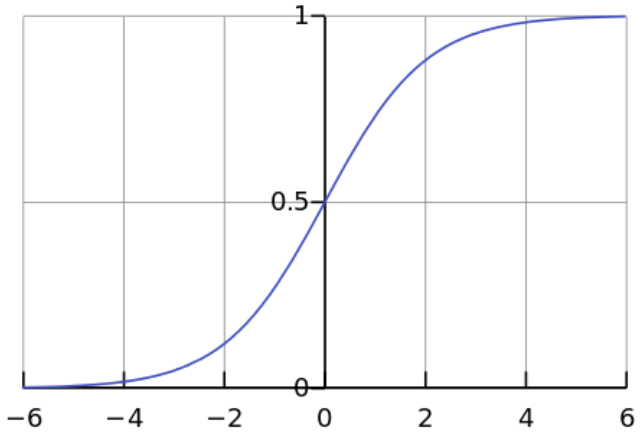
$$\frac{\partial P}{\partial w_1} = \frac{\partial P}{\partial z}\frac{\partial z}{\partial w_1} = (d-z)\frac{\partial z}{\partial w_1}$$

$$= (d-z)\frac{\partial z}{\partial p_2}\frac{\partial p_2}{\partial w_1} = (d-z)\frac{\partial z}{\partial p_2}\frac{\partial p_2}{\partial y}\frac{\partial y}{\partial w_1}$$

$$= (d-z)\frac{\partial z}{\partial p_2} w_2 \frac{\partial y}{\partial p_1}\frac{\partial p_1}{\partial w_1} = (d-z)\frac{\partial z}{\partial p_2} w_2 \frac{\partial y}{\partial p_1} x$$

Weight update..

$$\beta = \frac{1}{1+e^{-\alpha}} = \left(1+e^{-\alpha}\right)^{-1}$$



$$\frac{\partial \beta}{\partial \alpha} = \frac{e^{-\alpha}}{\left(1+e^{-\alpha}\right)^2} = \frac{1+e^{-\alpha}-1}{\left(1+e^{-\alpha}\right)^2}$$

$$= \frac{1}{\left(1+e^{-\alpha}\right)}\left[\frac{\left(1+e^{-\alpha}\right)}{\left(1+e^{-\alpha}\right)} - \frac{1}{\left(1+e^{-\alpha}\right)}\right] = \beta(1-\beta)$$

Weight update…

$$\frac{\partial z}{\partial p_2} = z(1-z)$$

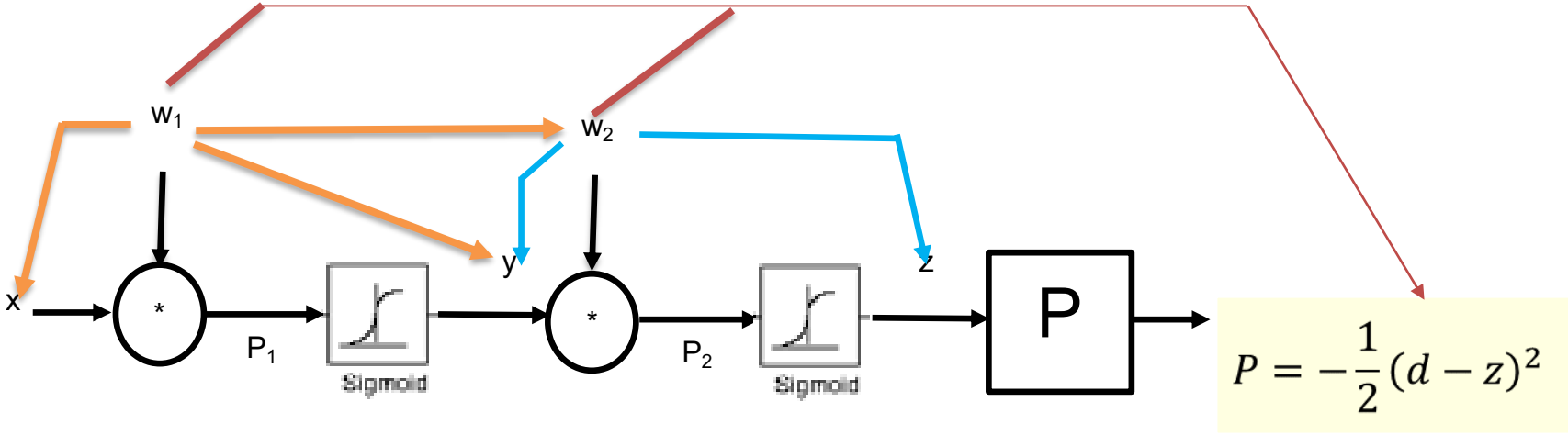
$$\frac{\partial y}{\partial p_1} = y(1-y)$$

$$\frac{\partial P}{\partial w_2} = (d-z)\frac{\partial z}{\partial p_2} y = (d-z)z(1-z)y$$

$$\frac{\partial P}{\partial w_1} = (d-z)\frac{\partial z}{\partial p_2} w_2 \frac{\partial y}{\partial p_1} x$$

$$= (d-z)z(1-z)w_2 y(1-y)x$$

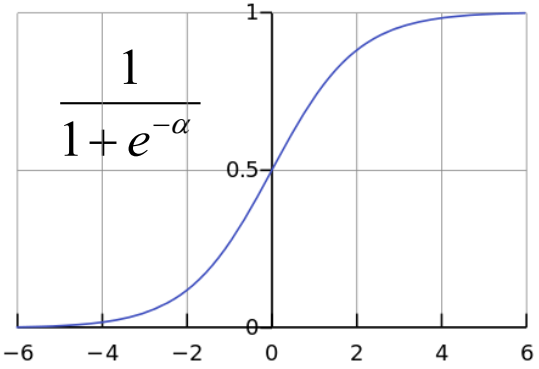
Weight update…



$$\frac{\partial P}{\partial w_2} = (d-z)\frac{\partial z}{\partial p_2}\, y = (d-z)z(1-z)y$$

$$\frac{\partial P}{\partial w_1} = (d-z)\frac{\partial z}{\partial p_2}\, w_2\, \frac{\partial y}{\partial p_1}\, x$$

$$= (d-z)z(1-z)w_2 y(1-y)x$$



Weight update…

- Finally, the Δw is

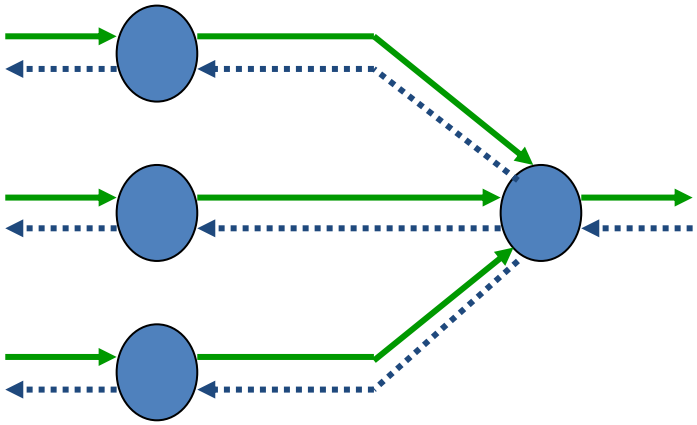
$$\Delta w = \eta\left(\frac{\partial P}{\partial x} i + \frac{\partial P}{\partial y} j\right) = \eta\left(\frac{\partial P}{\partial w_1} + \frac{\partial P}{\partial w_2}\right)$$

$$\Delta \text{w} = \eta \left((\text{d-z})\text{z}(1\text{-z}) w_2 y(1-y)x \right) + (\text{d}-\text{z})\text{z}(1-\text{z}))$$

Then updates the weight with Δw

Training Algorithm: Back-propagation

- Back-propagation algorithm learns in the same way as single perceptron.
- Back propagation adjusts the weights of the NN to minimize the network total errors.



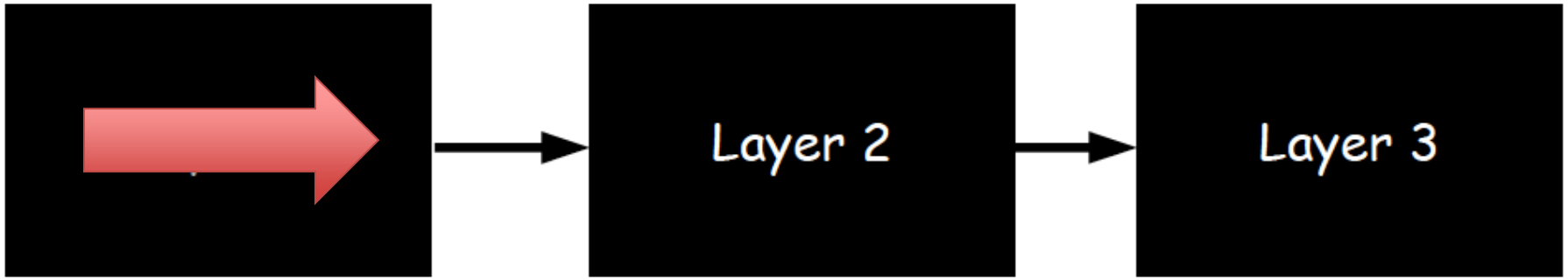
Network activation& error cal. Forward Step

Error propagation

Backward Step to adjust weights

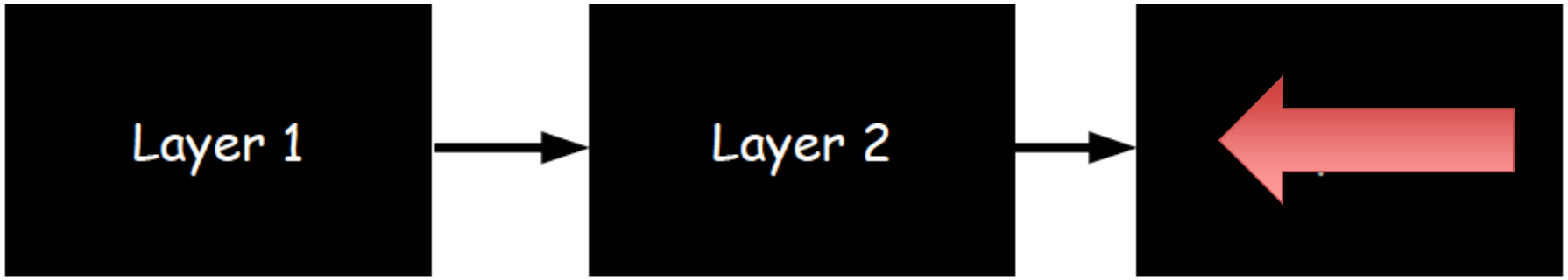
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]



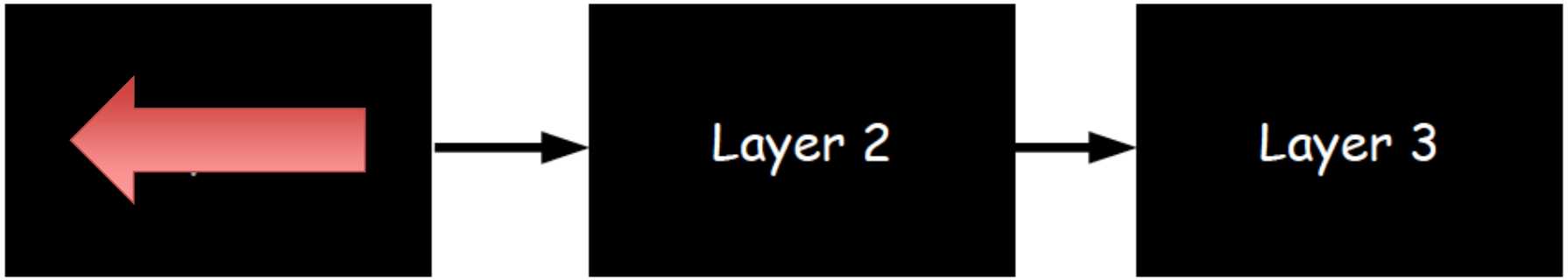
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]



Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]
- Step 3: Use gradient to update parameters



$$\theta \leftarrow \theta - \eta \frac{dL}{d\theta}$$

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Gradient decent with Back-prop Algorithm

1. Initialize all weights to small random values
2. REPEAT until done:

   a) For each with w_{ij} set $\Delta w_{ij} := 0$

   b) For each data point points $(\mathbf{x}, \mathbf{t})^p$

      1. set input units to **x**
      2. compute value of output units
      3. For each weight w_{ij} set $\Delta W_{ij} := \Delta W_{ij} + (t_i - y_i)\, x_i$

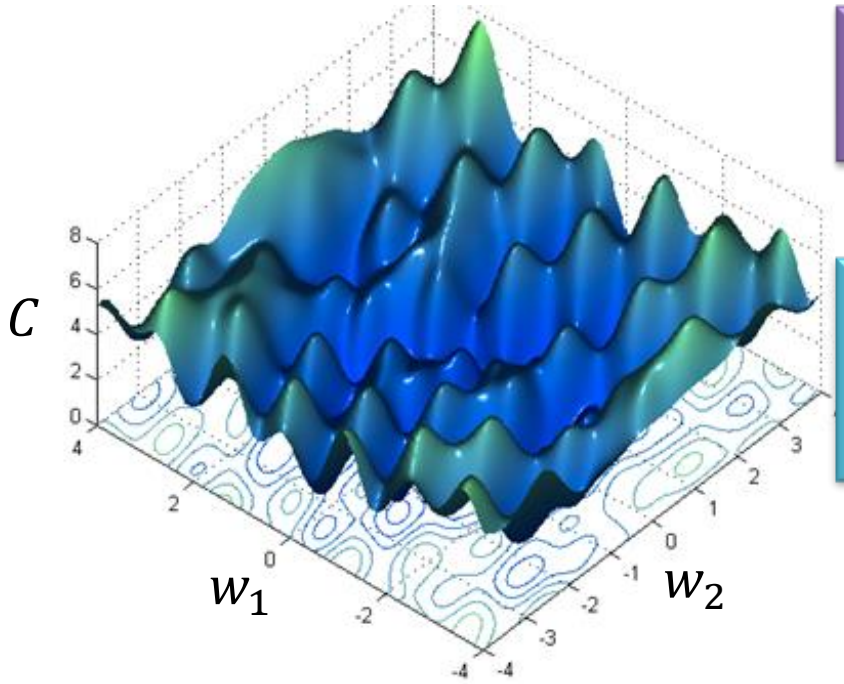
   c) For each weight w_{ij} set $w_{ij} := w_{ij} + \Delta w_{ij}$

   or $w_{ji}(t+1) = w_{ji}(t) + \alpha \Delta w_{ji}(t)$

3. The algorithm terminates once we are at, or sufficiently near to, the minimum of the error function, where G = 0.
4. We say then that the **network has converged.**

Local Minima

- Gradient descent never guarantee global minima



Different initial point θ^0

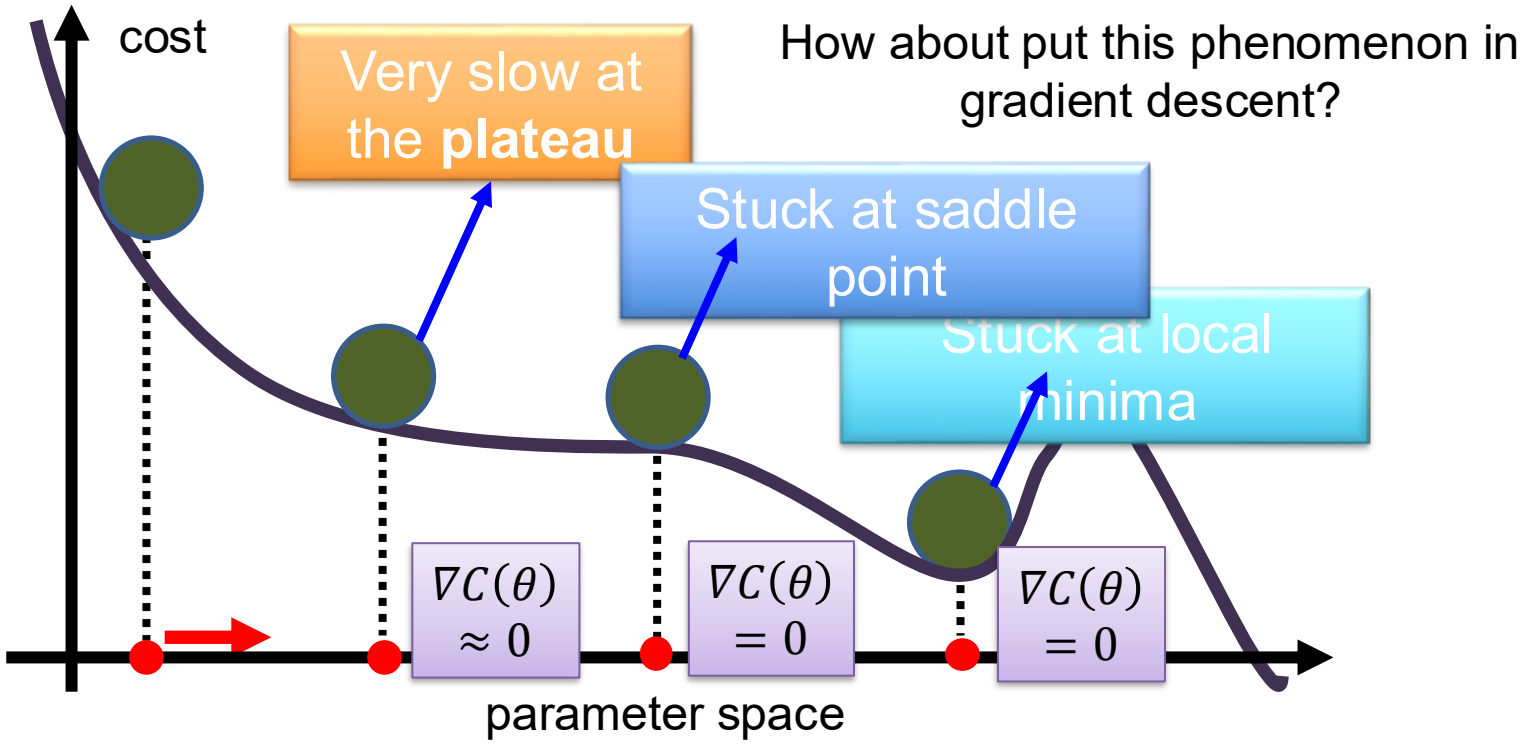
Reach different minima, so different results

Who is afraid of Non-Convex Loss Functions?

http://videolectures.net/eml07_lecun_wia/

Besides local minima ……

- The momentum method by (Polyak, 1964)

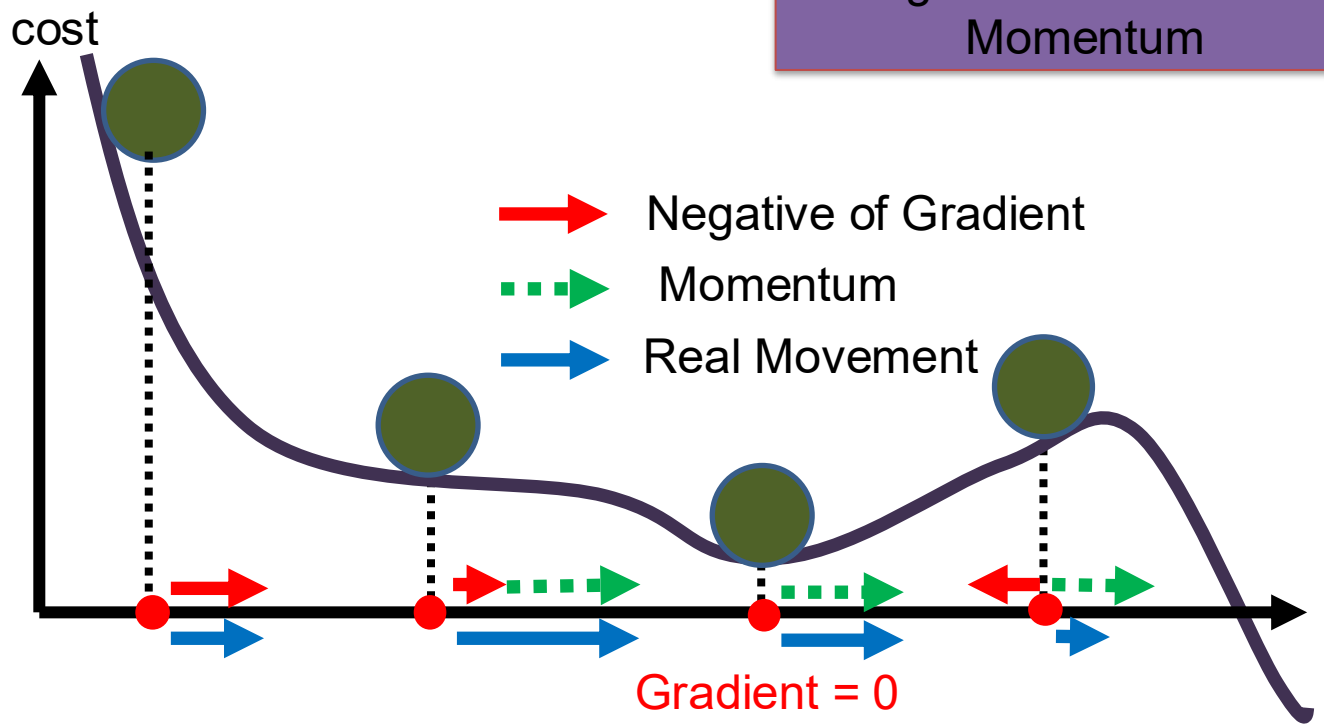


Momentum

Still not guarantee reaching global minima, but give some hope ……

Movement =

Negative of Gradient + Momentum



Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." *International conference on machine learning*. 2013.

Why Momentum?

- Accelerate the training process
- Prevent to stuck in local minima

- Momentum values: $\gamma \in (0,1]$
  - Generally, γ is set to 0.5 until the initial learning stabilizes and then is increased to 0.9 or higher

Learning approach with NNs: sample by sample

- Problem with individual sample learning
  - Learning with individual sample over the training sets converge very well to local optima (minima).
  - In practice computing cost and gradient computation for the entire training set is very slow.
  - Not suitable for a single machine if the **dataset is too big to fit to the memory**

- **Solution**: Batch learning

Learning approach with NNs: Batch learning

- Stochastic Gradient Descent (SGD) can **overcome computational cost and still lead to fast convergence.**
- Instead of learning whole training dataset at a time by sample, take a batch of images for one iteration.

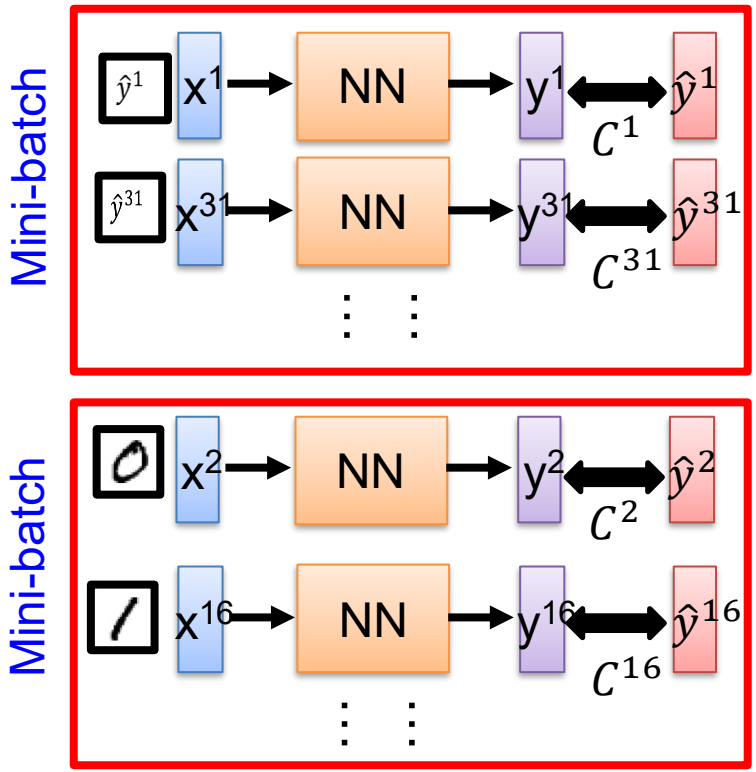
number of batch= Total training samples / Number of samples per batch

- We define batch-size during the model training
- Batch learning user for **data parallelism**

Mini-batch

Faster

Better!



- Randomly initialize θ^0
- Pick the 1st batch

$$C = C^1 + C^{31} + \cdots$$
$$\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$$

- Pick the 2nd batch

$$C = C^2 + C^{16} + \cdots$$
$$\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$$

⋮

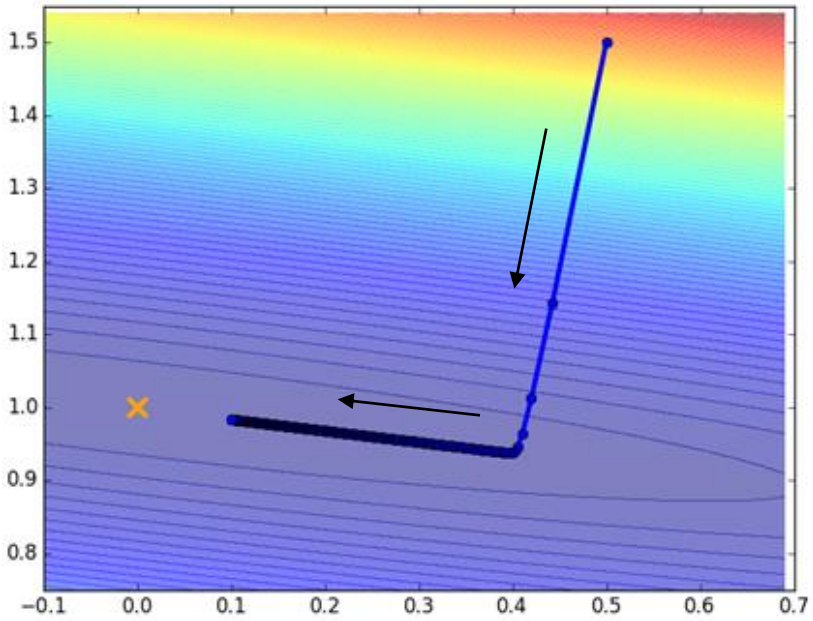
- Until all mini-batches have been picked

one epoch

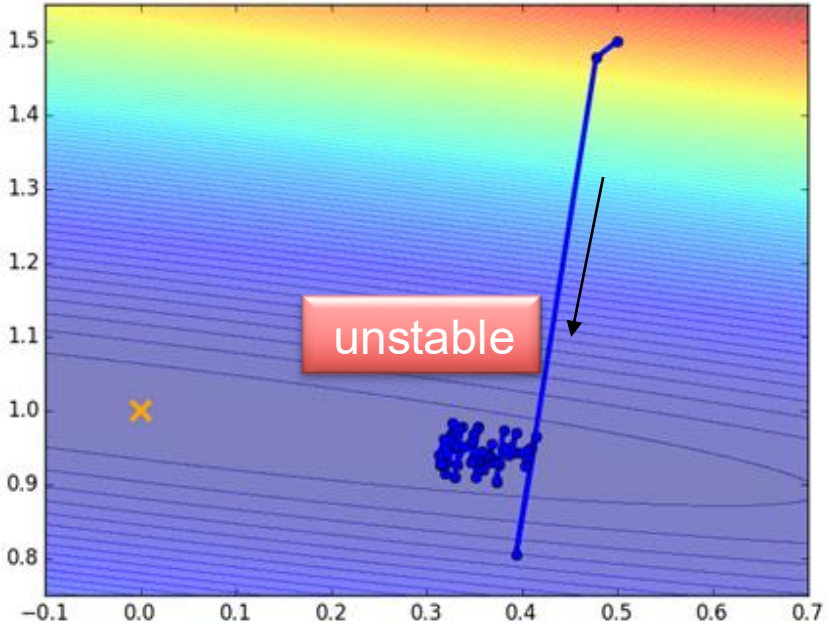
Repeat the above process

Mini-batch

Original Gradient Descent



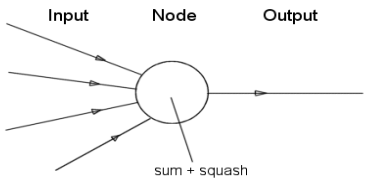
With Mini-batch: Stochastic Gradient Descent (SGD)



The colors represent the total C on all training data.

Learning rate with mini-batch

- All the neurons in the MLP should learn ideally of the same rate
- Last layers gradient is larger than the layers in the front end of the network
- η should be assign with smaller value in the last layers than in the front layers
- Neurons with more inputs should have smaller learning rate compare to few inputs
- LeCunn (1993) : $\eta \infty \frac{1}{\sqrt{\#\ syp.connections\ to\ a\ neuron}}$



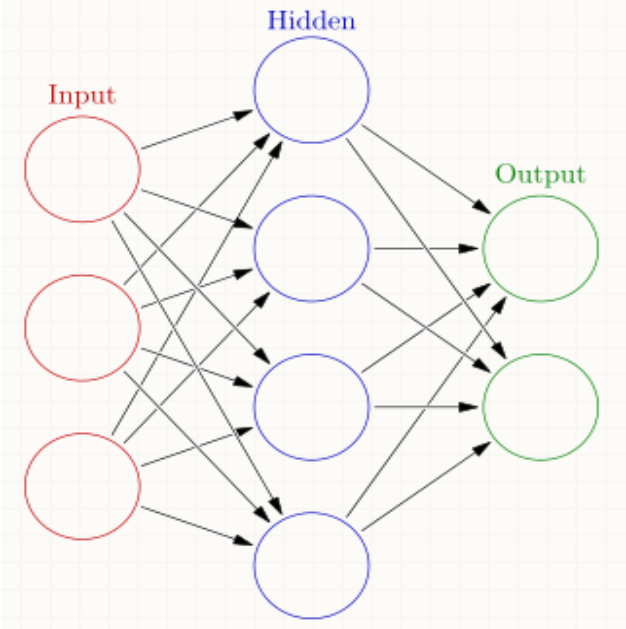
Stopping criterions

- Total **mean squared error** change or loss:
  - Back-prop is considered to have converged when the absolute rate of change in the average squared error per epoch is sufficiently small (in the range [0.1, 0.01]).
  - Maximum number of iterations or epochs (N)

- **Generalization** based criterion:
  - After each epoch, the NN is tested for generalization.
  - If the generalization performance is satisfactory then stop.
  - If this stopping criterion is used, the part of the training set used for testing the network generalization will not used for updating the weights.

Computational parameters for DNN

- Every connection that is learned in a neural network is called parameter.

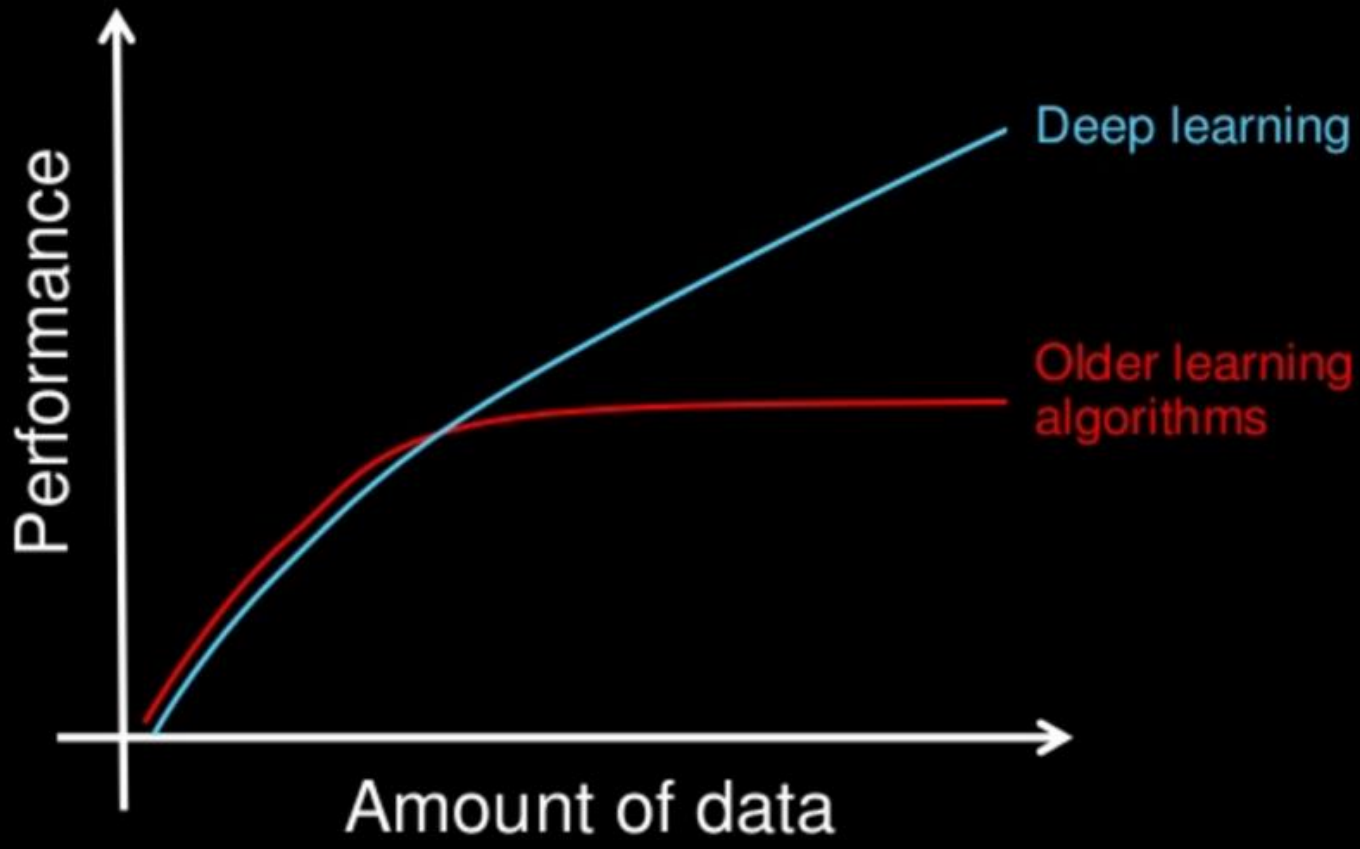


Without bias: (3*4)+(4*2)=20

With bias: (4*4)+(5*2)=26

If network is not fully connected **you can count the connections**.

Why deep learning



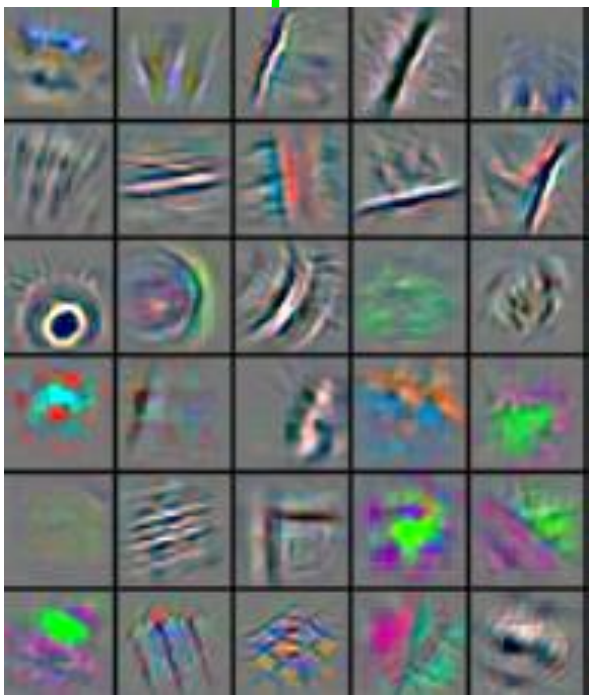
How do data science techniques scale with amount of data?

Properties of Deep Learning approaches

- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations
- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction
- Distributed Representations , Scalability, and Genericity
  - No single neuron “encodes” everything
  - Groups of neurons work together

Deep Learning = Hierarchical Compositionality

Low-Level Feature → Mid-Level Feature → High-Level Feature → Trainable Classifier → "car"



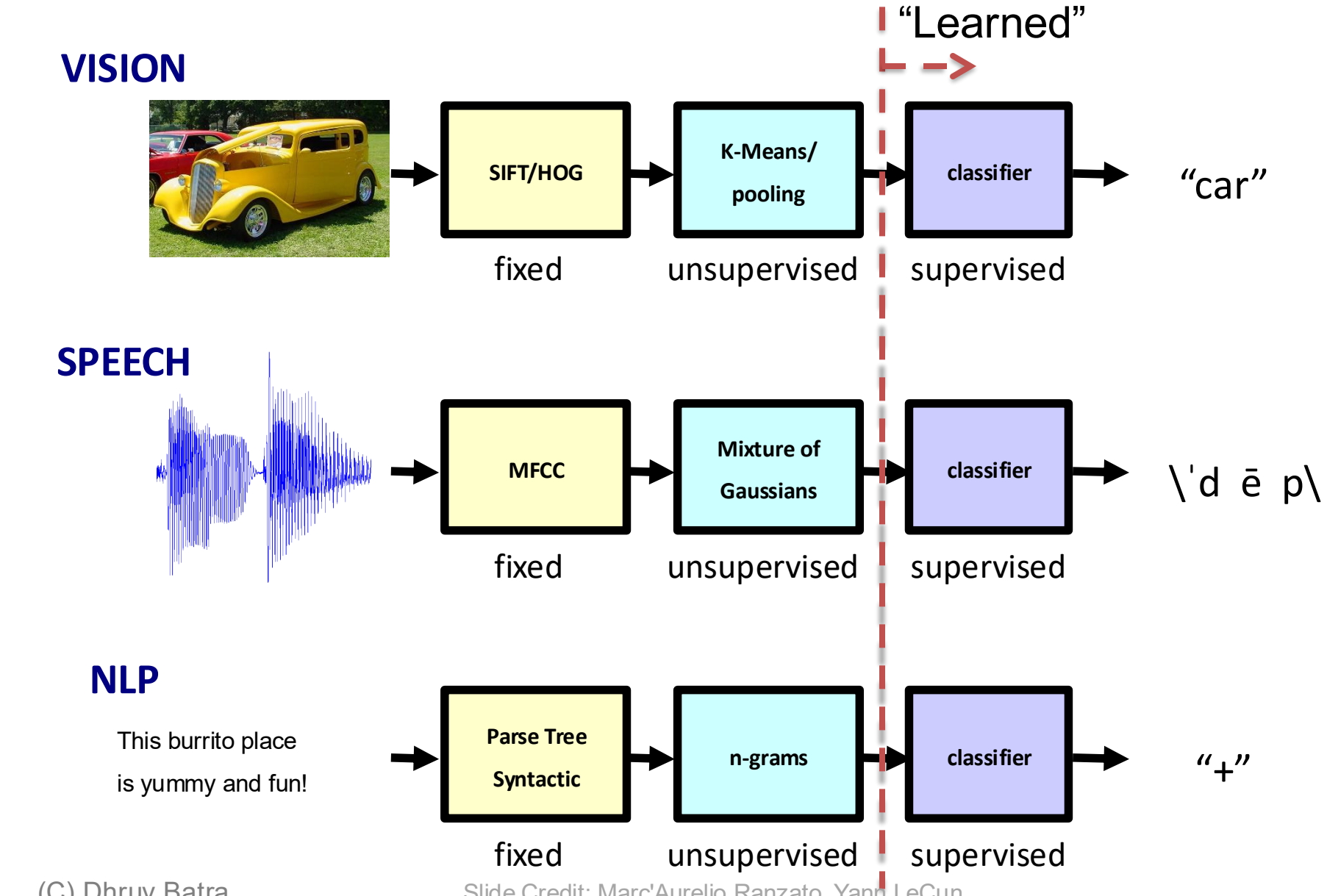
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Slide Credit: Marc'Aurelio Ranzato, Yann LeCun

Properties of Deep (Machine) Learning approaches

- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations
- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction
- Distributed Representations , Scalability, and Genericity
  - No single neuron “encodes” everything
  - Groups of neurons work together

Deep Learning = End-to-End Learning



Properties of Deep (Machine) Learning approaches

- (Hierarchical) Compositionality
  - Cascade of non-linear transformations
  - Multiple layers of representations

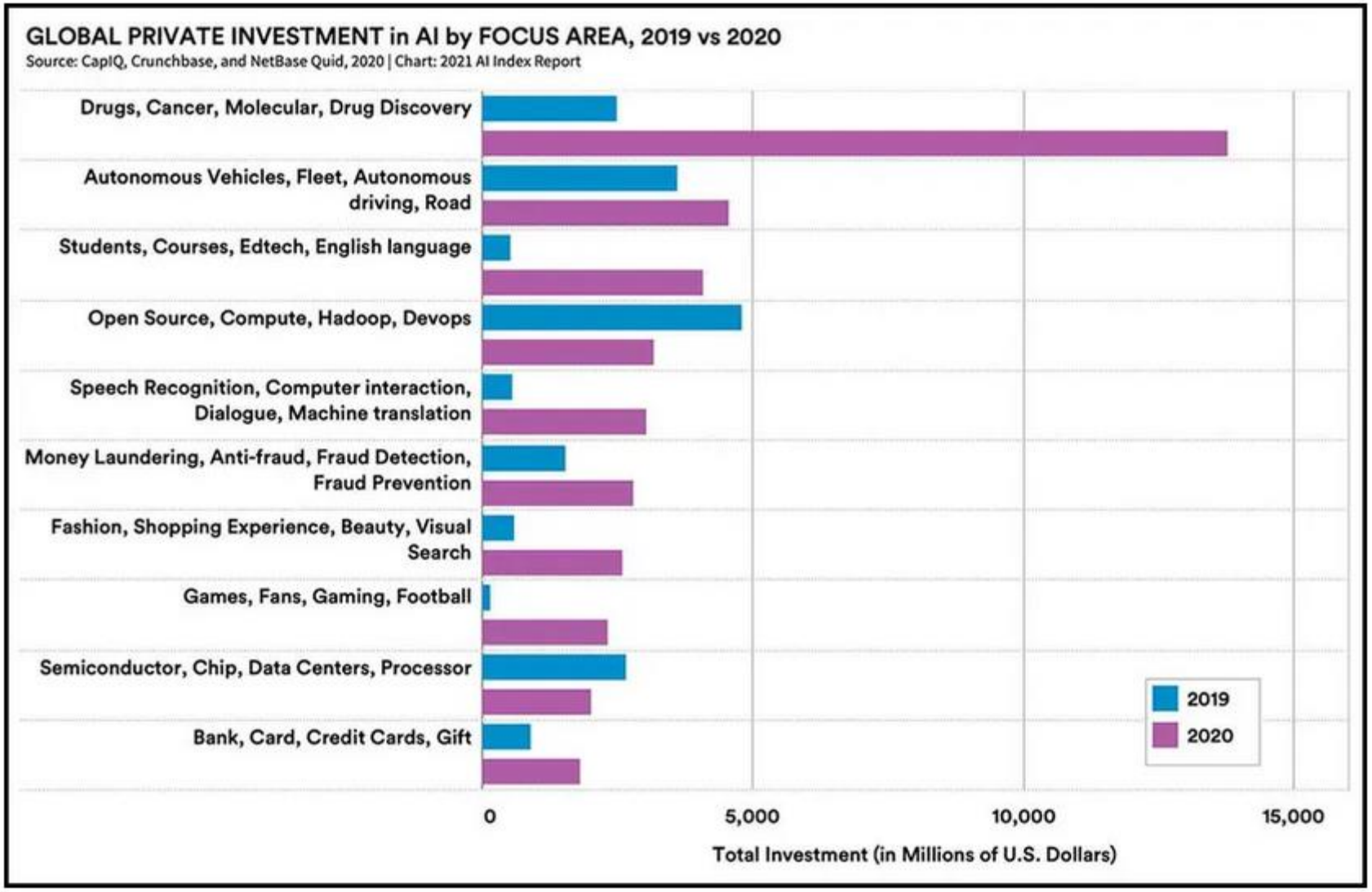
- End-to-End Learning
  - Learning (goal-driven) representations
  - Learning to feature extraction

- Distributed **Representations , Scalability, and Genericity**
  - No single neuron “encodes” everything
  - Groups of neurons work together

One Model To Learn Them All

Kaiser, Lukasz, Aidan N. Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. "One model to learn them all." *arXiv preprint arXiv:1706.05137*(2017).

Deep learning applications

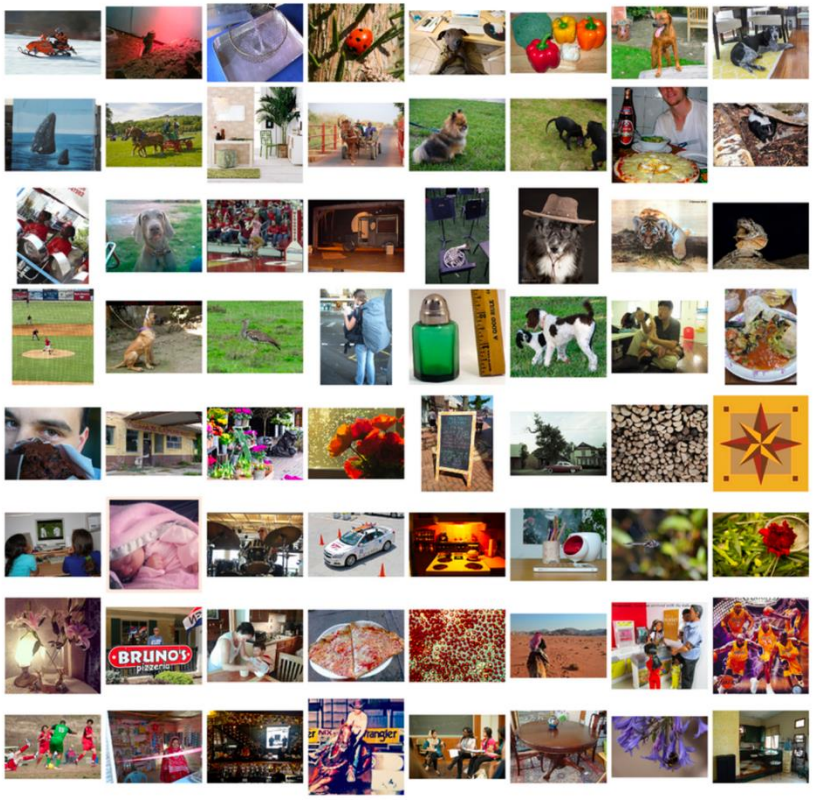


Source: https://spectrum.ieee.org/the-state-of-ai-in-15-graphs/9-the-startup-frenzy-is-over

Image Classification

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

21,841 object classes **14+ M images**



https://www.image-net.org/update-mar-11-2021.php

Image Classification

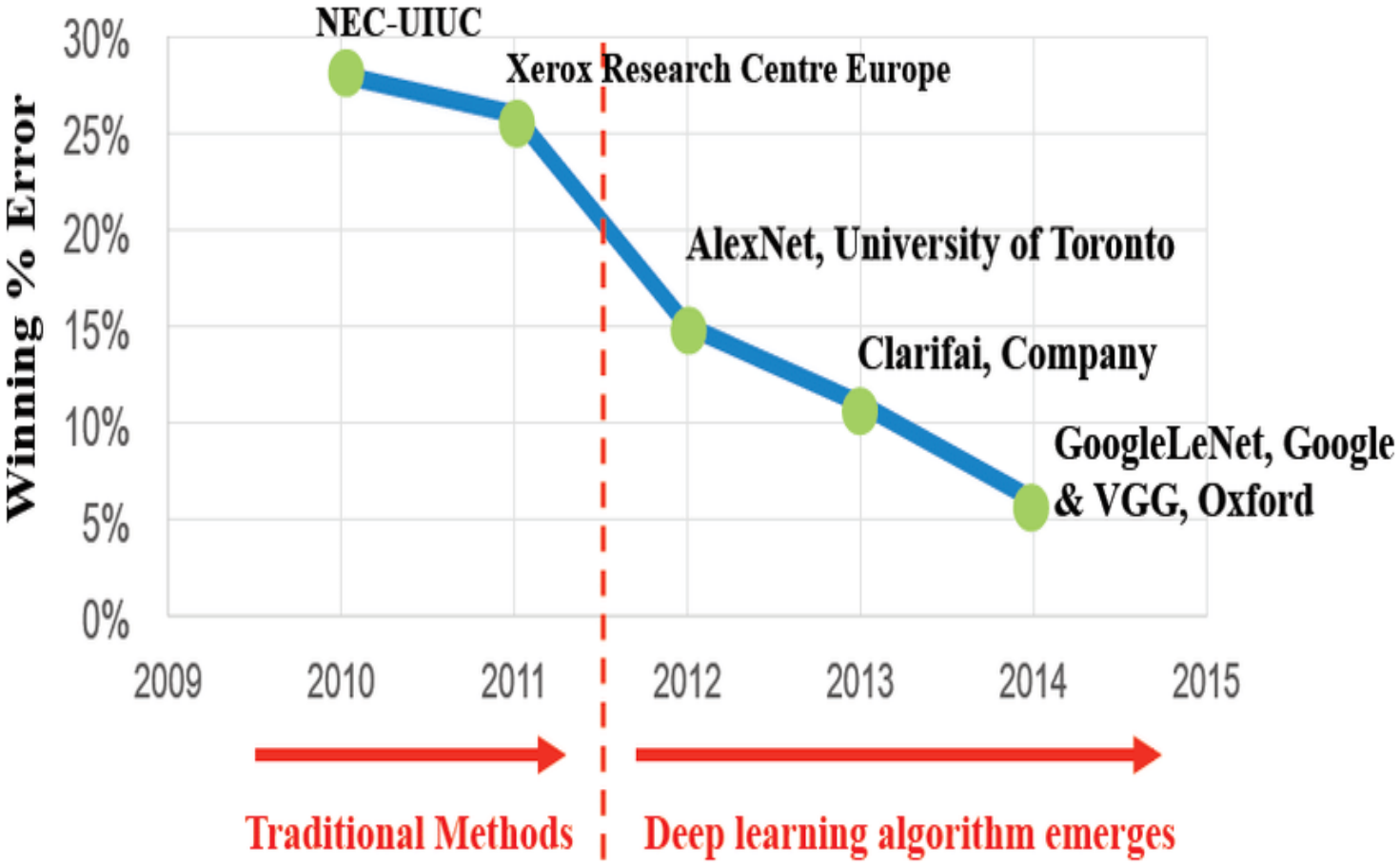
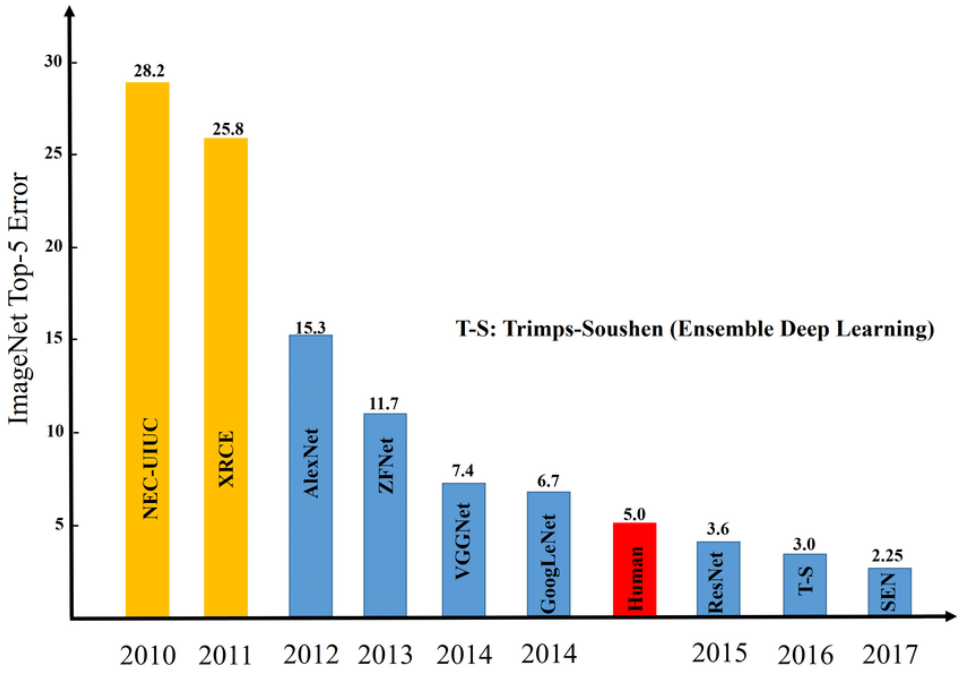
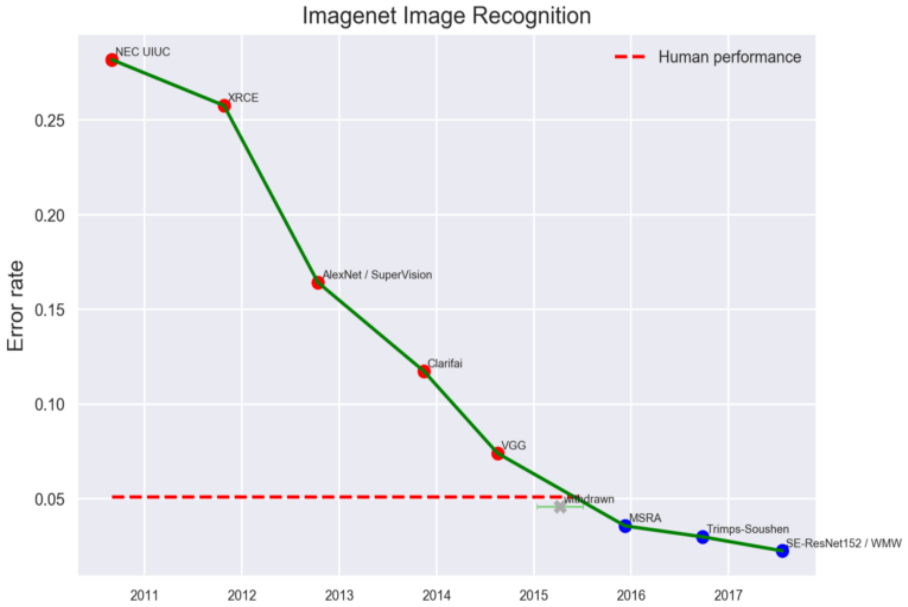


Image classification





https://paperswithcode.com/sota/image-classification-on-imagenet

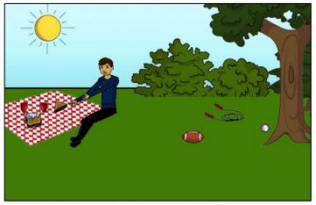
Tasks are getting bolder

Image captioning: A group of young people playing a game of Frisbee

Vinyals et al., 2015

What color are her eyes?
What is the mustache made of?

How many slices of pizza are there?
Is this a vegetarian pizza?



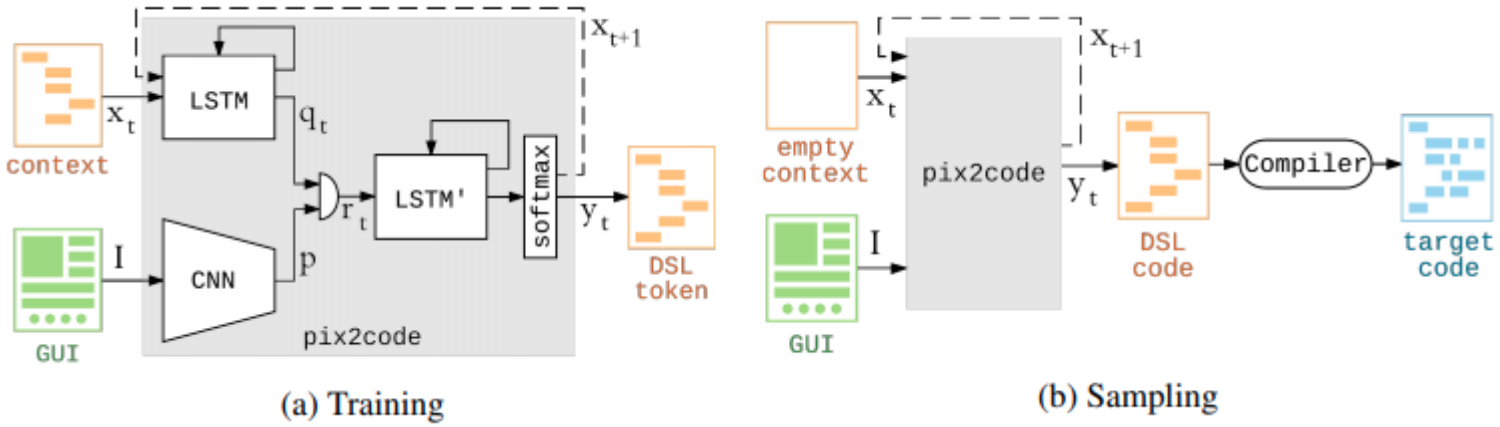
Is this person expecting company?
What is just under the tree?

Does it appear to be rainy?
Does this person have 20/20 vision?

Visual Question Answering (VQA) : Antol et al., 2015

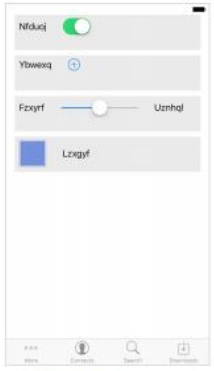
Das et al., 2017

Pix2code



Overview of the pix2code model architecture

- Transforming a graphical user interface screenshot into computer code
- Over 77% of accuracy for three different platforms (i.e. iOS, Android and web-based technologies)



(a) iOS GUI screenshot

```
stack {
    row {
        label, switch
    }
    row {
        label, btn-add
    }
    row {
        label, slider, label
    }
    row {
        img, label
    }
}
footer {
    btn-more, btn-contact, btn-search, btn-download
}
```

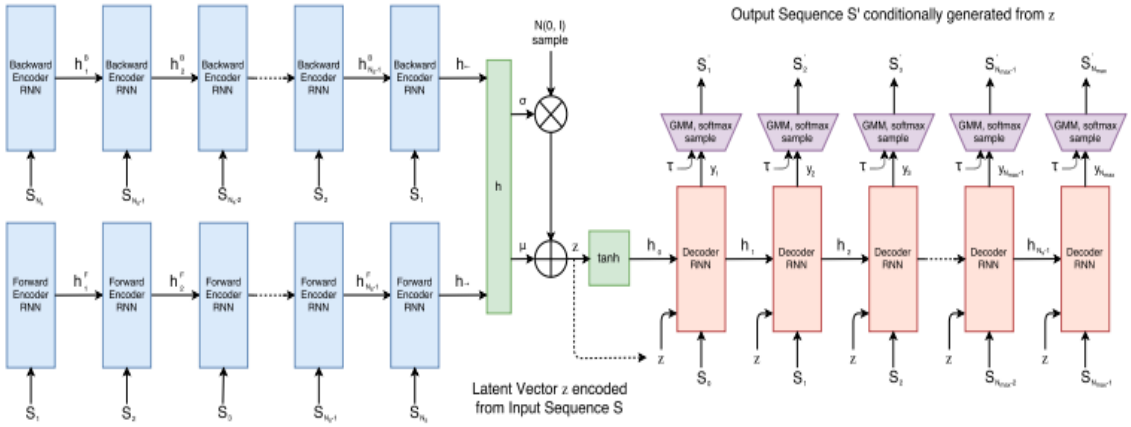
(b) Code describing the GUI written in our DSL

An example of a native iOS GUI written in our markup-like DSL.

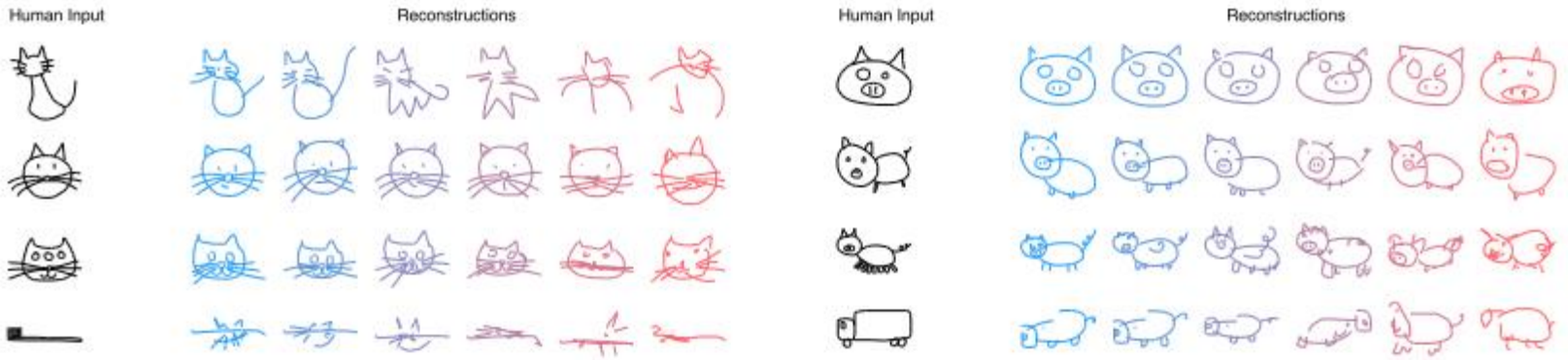
Beltramelli, Tony. "pix2code: Generating code from a graphical user interface screenshot." *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*. 2018.

SketchRNN: teaching a machine to draw

- A recurrent neural network (RNN) able to construct stroke-based drawings of common objects



Sketch-RNN



Conditional generation of cats (left) and pigs (right).

Ha, David, and Douglas Eck. "A neural representation of sketch drawings." *arXiv preprint arXiv:1704.03477* (2017).

Language understanding

- Recognize phrases and sentences being spoken by a talking face, with or without the audio

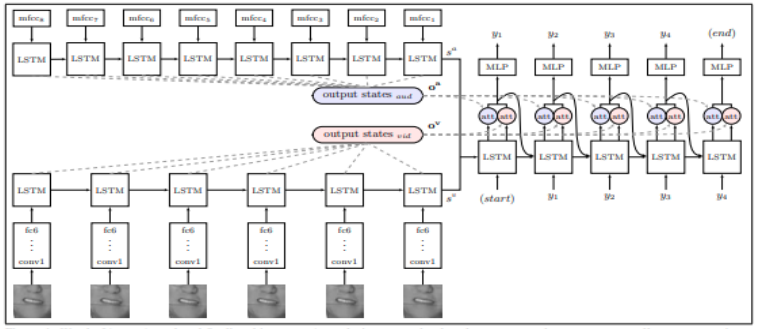


Figure 1. *Watch, Listen, Attend and Spell* architecture. At each time step, the decoder outputs a character y_t, as well as two attention vectors. The attention vectors are used to select the appropriate period of the input visual and audio sequences.

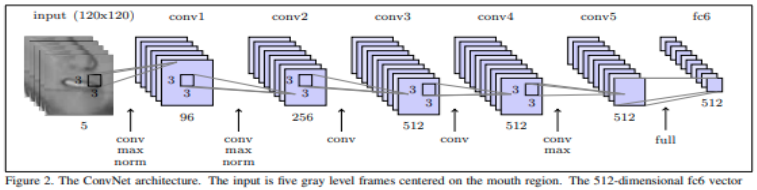
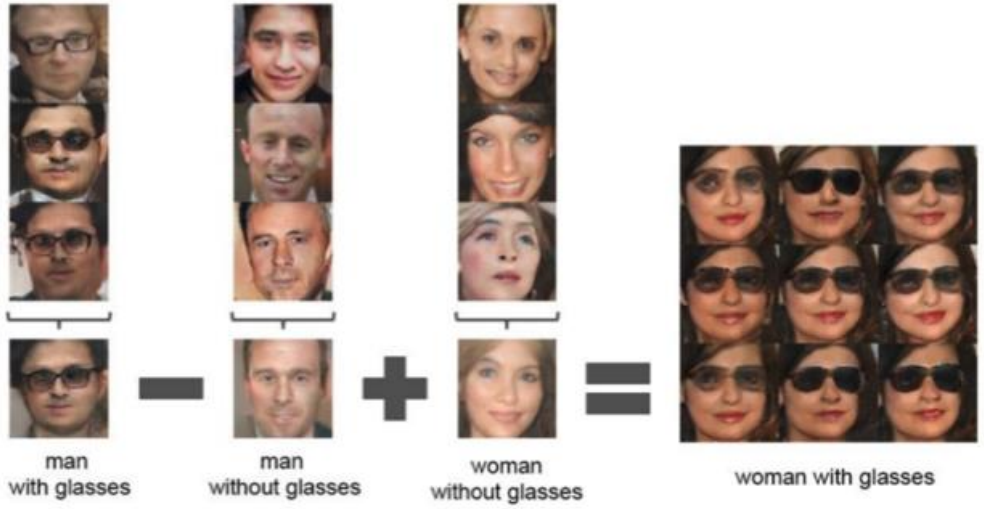


Figure 2. The ConvNet architecture. The input is five gray level frames centered on the mouth region. The 512-dimensional fc6 vector forms the input to the LSTM.

Chung JS, Senior A, Vinyals O, Zisserman A. Lip reading sentences in the wild. In2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017 Jul 21 (pp. 3444-3453). IEEE.

Application of generative models

- GANs Generated images



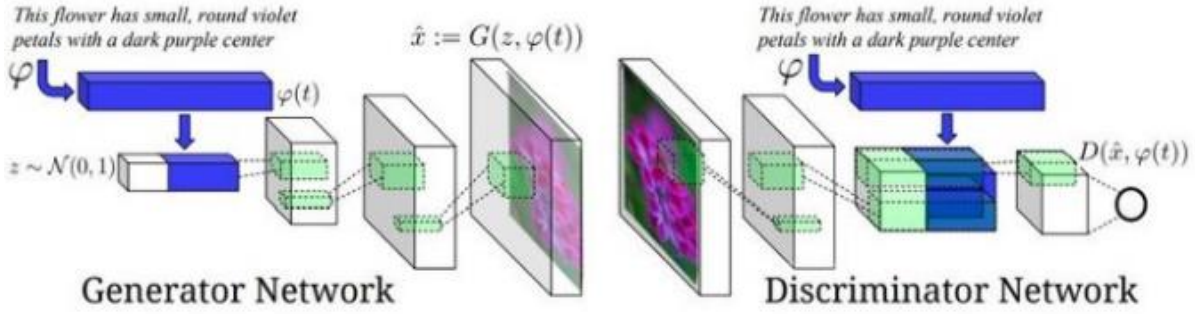
StyleGAN results

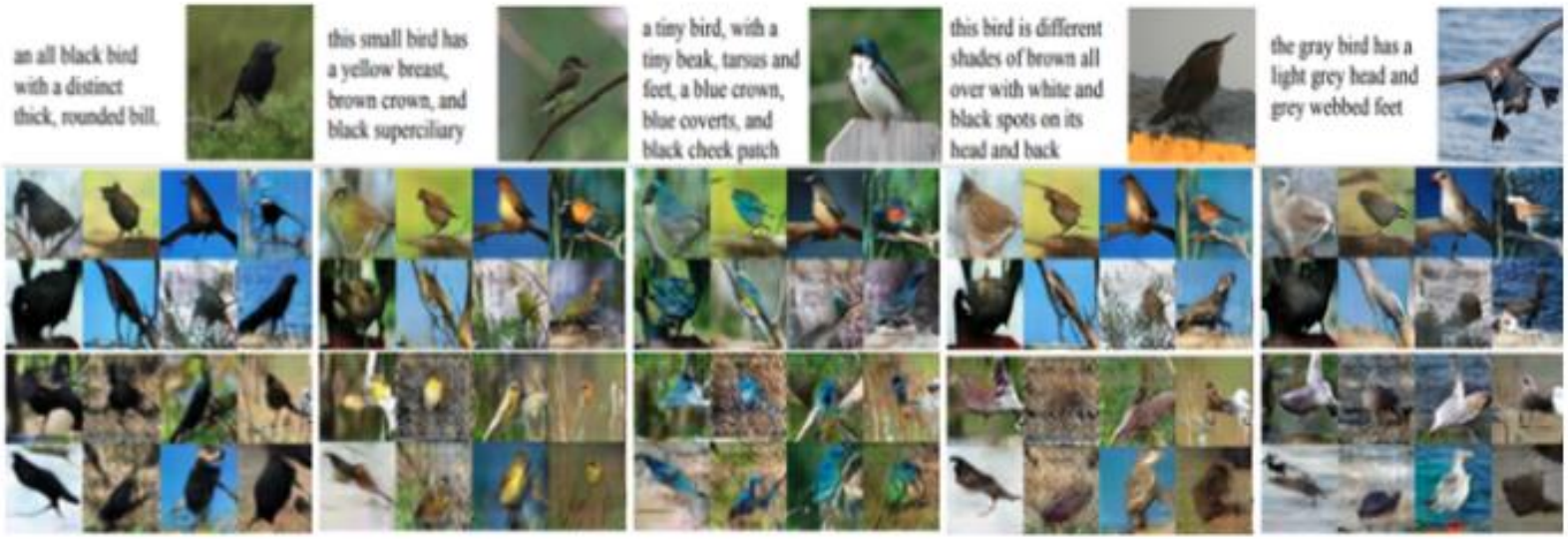
Picture: *These people are not real – they were produced by our generator that allows control over different aspects of the image.*

Karras, Tero, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401-4410. 2019.

Synthesization of an image from a text description

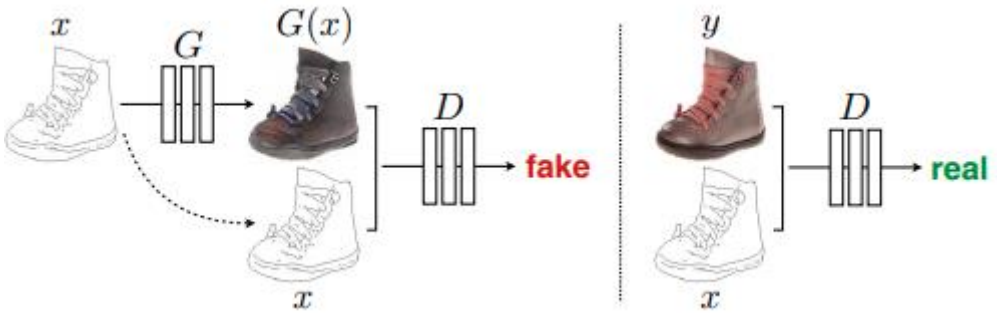
- Text2Image:

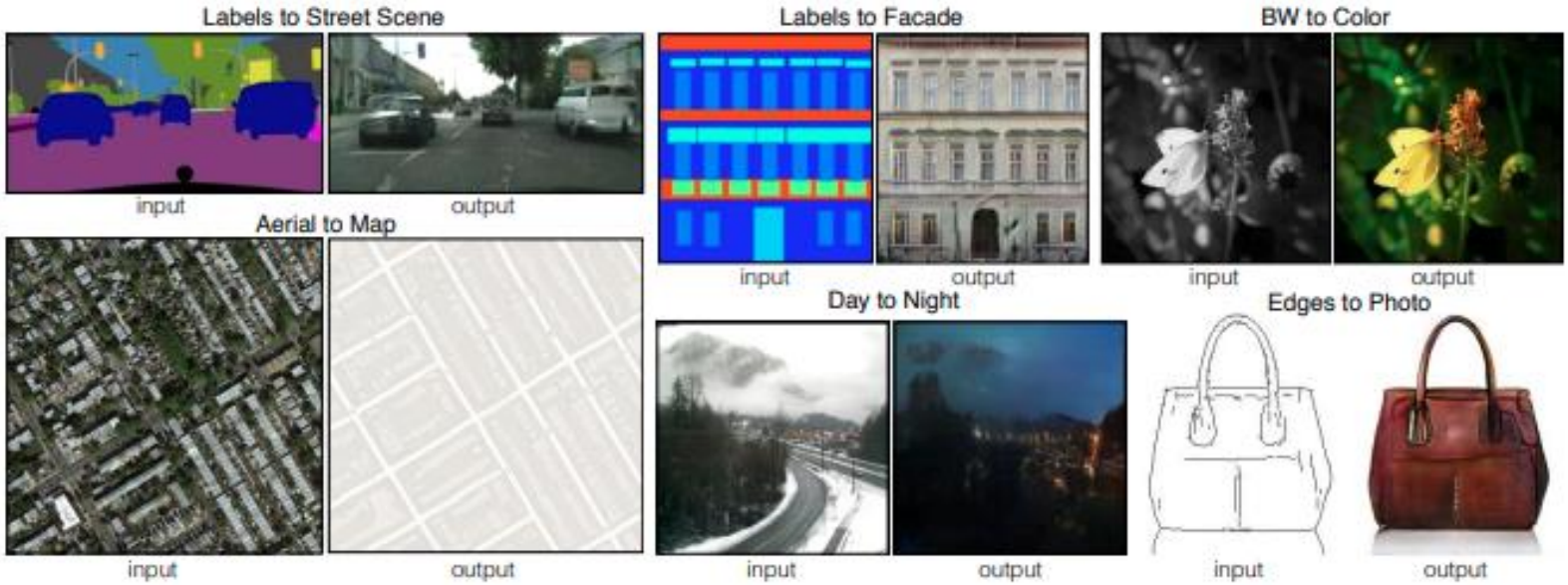




Pix2pix

- Image2Image…





Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-image translation with conditional adversarial networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125-1134. 2017.

Google DeepMind AlphaGo vs Lee Sedol

AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol

DeepMind's artificial intelligence astonishes fans to defeat human opponent and offers evidence computer software has mastered a major challenge

The world's top Go player, Lee Sedol, lost the final game of the Google DeepMind challenge match. Photograph: Yonhap/Reuters

Google DeepMind's AlphaGo program triumphed in its final game against South Korean Go grandmaster Lee Sedol to win the series 4-1, providing further evidence of the landmark achievement for an artificial intelligence program.

https://www.youtube.com/watch?v=WXuK6gekU1Y&t=899s

DeepMind solves protein folding | AlphaFold 2

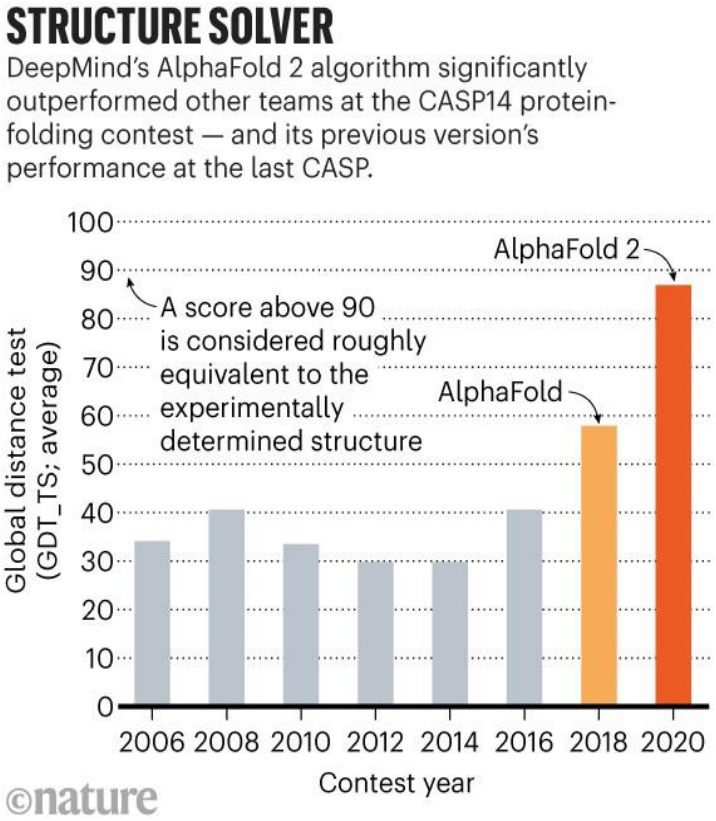
What happened?

DeepMind's AlphaFold2 solves protein folding **(50 years old grand challenge)**

→ "Solves" = Achieves 87+ GDT of CASP competition

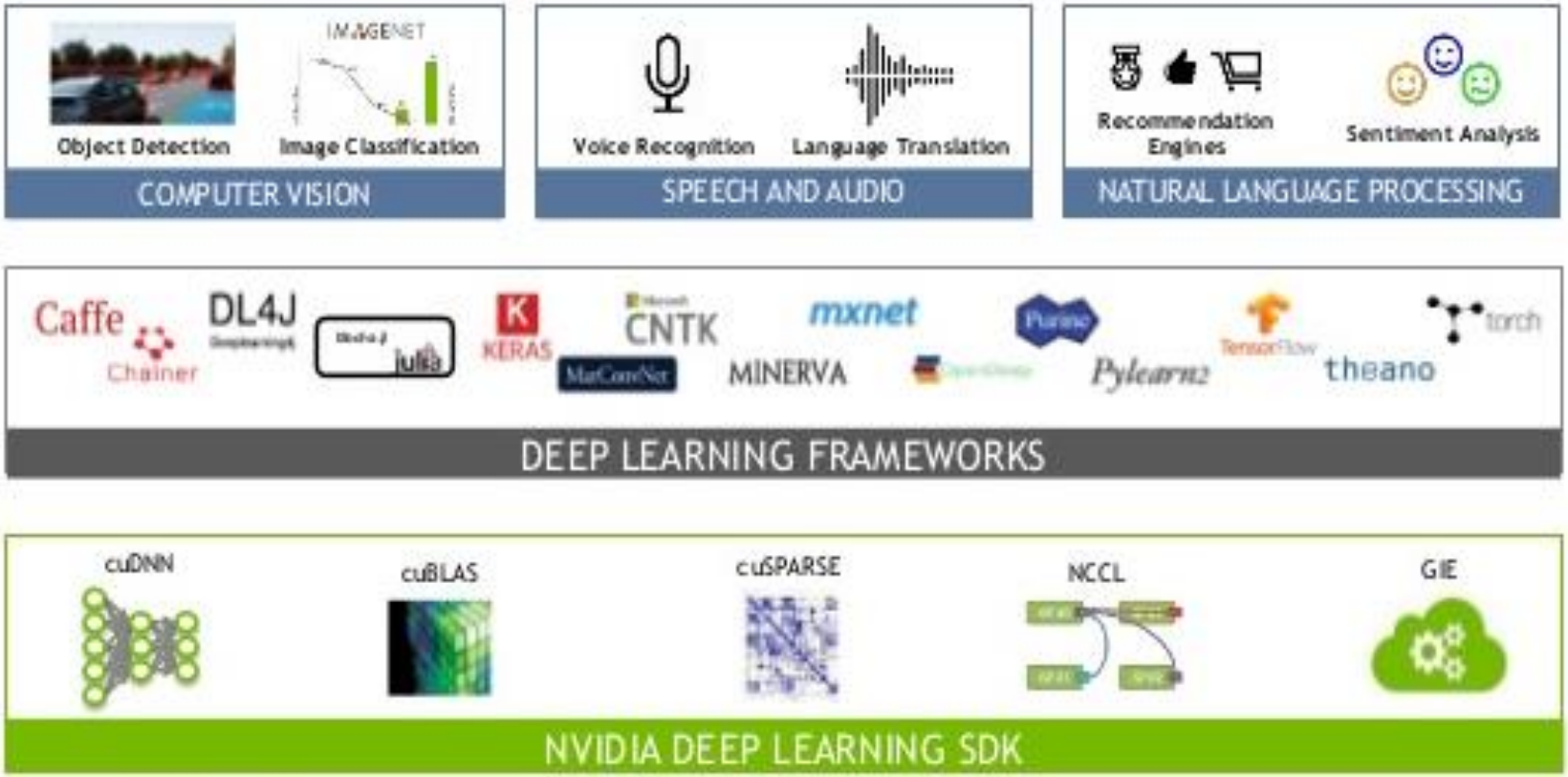
How big is this accomplishment?

- Biggest advancement is **structural biology** of the past 20+ years
- Biggest advancement in artificial intelligence of the past 20+ years
  - ImageNet moment (AlexNet)
- **Selected : First Nobel Prize for machine learning model**



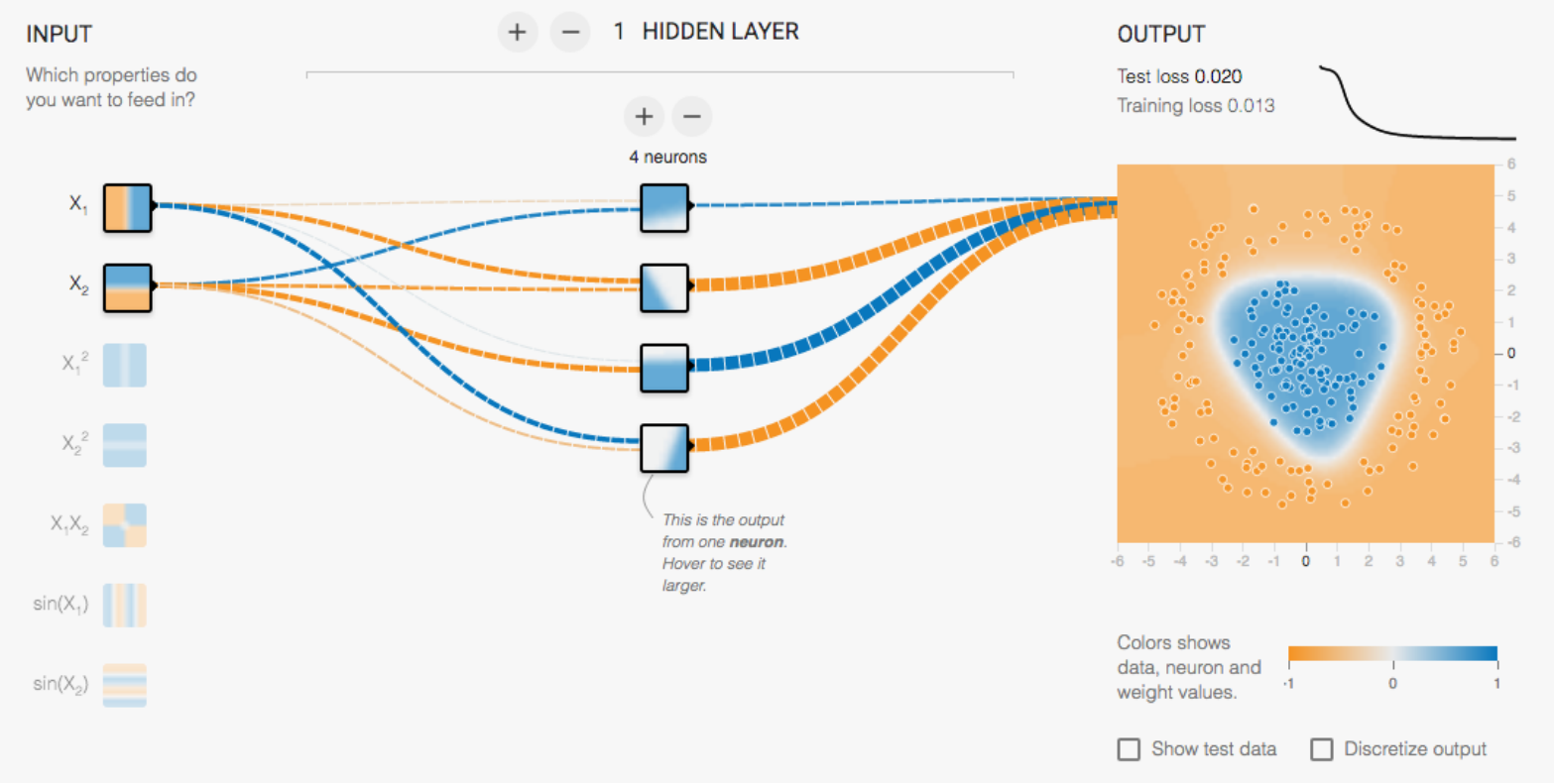
Powering the Deep Learning Ecosystem

NVIDIA SDK Accelerates Every Major Framework



oper.nvidia.com/deep-learning-software

Multi-Layer Networks Demo



http://playground.tensorflow.org/

Summary

- What is NNs?
- The NN optimization with back-propagation (BP)
- Understanding Neural Networks Through Visualization
  - Impact of inputs (features) and hidden layers
- Mathematical model and BP approach for a NN
- Momentum, batch learning method, and network parameters
- Why deep learning and its applications

- What's next:
  - Computational graph
  - Efficient gradient computation methods