Artificial Intelligence: Knowledge Representation: Neural Networks

CPSC 433: Artificial Intelligence
Fall 2022

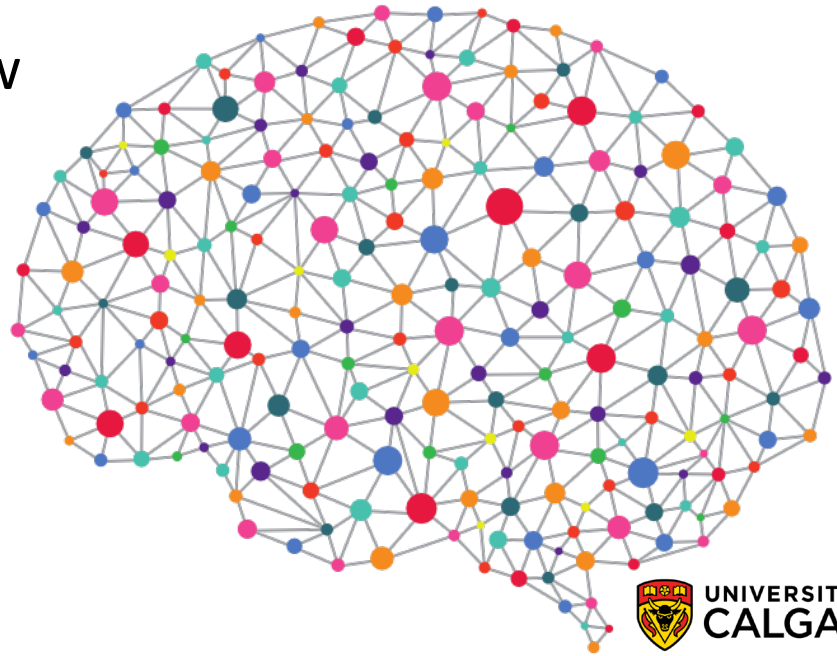
Jonathan Hudson, Ph.D
Assistant Professor (Teaching)
Department of Computer Science
University of Calgary

Monday, October 24, 2022

UNIVERSITY OF CALGARY

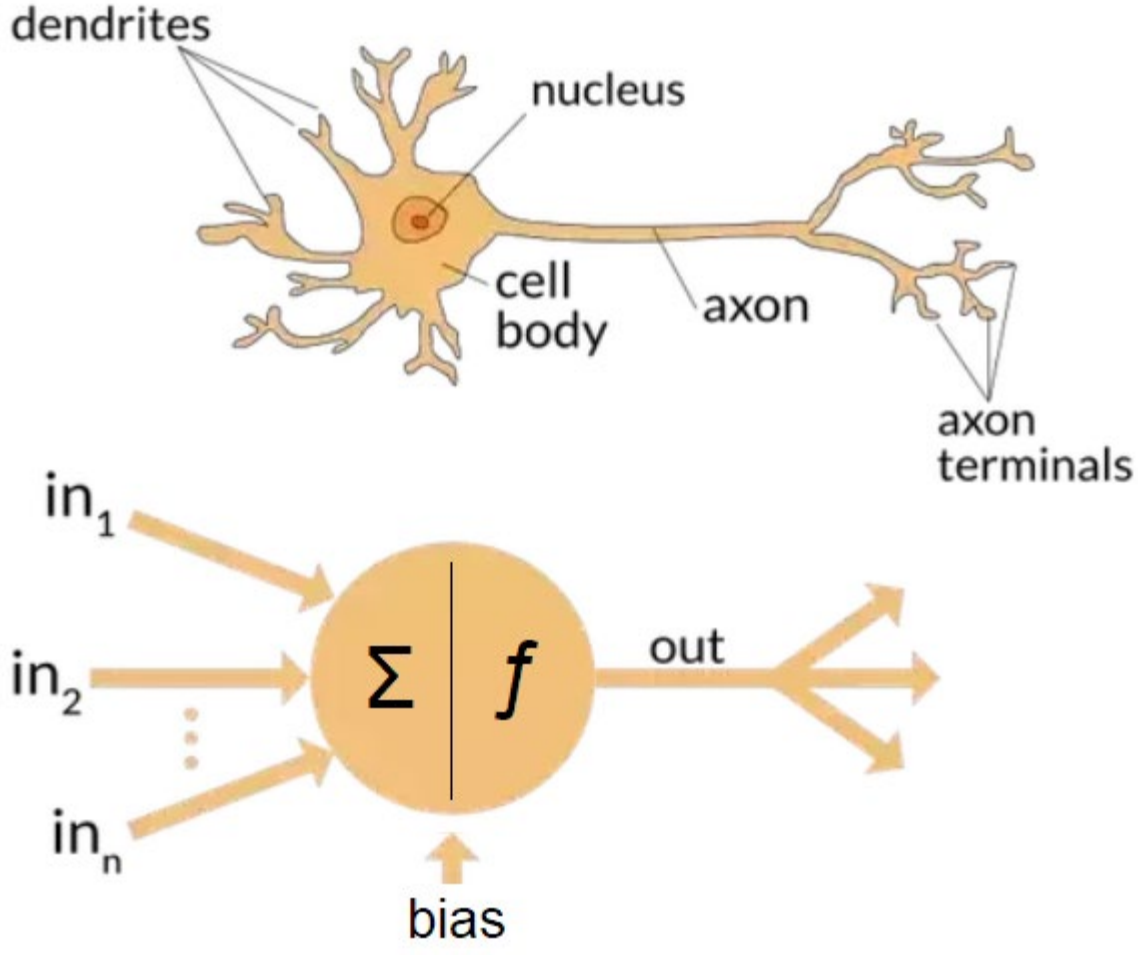
What are neural networks?

- Inspired by the Human Brain.
- The human brain has about 86 Billion neurons and requires 20% of your body's energy to function.
- These neurons are connected to between 100 Trillion to 1 Quadrillion synapses!
- Deep learning neural networks really popular right now
- Connectionist method
  - Make network, train, hope result is useful



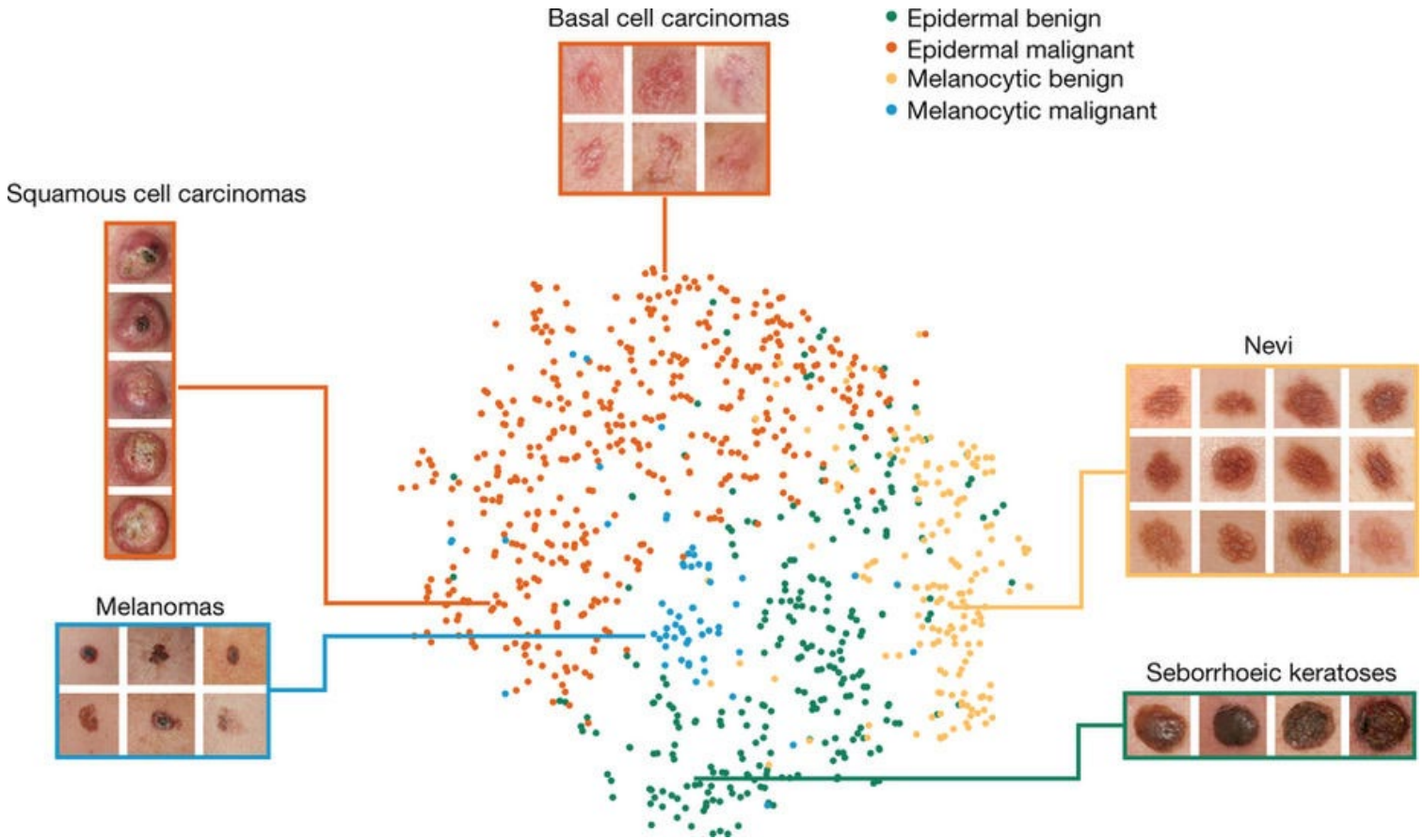
Neuron Model of Connections

- Developed to mimic the human neural system (in the brain) and its processing capabilities
- Decentralized knowledge representation and processing
  ☞ hopefully very efficient
- Simple components, the intelligence is in the connections



Examples

Examples: Classify skin cancer



Dermatologist-level classification of skin cancer with deep neural networks (Esteva et al., Nature 2017)

Examples: Neural Style Translation

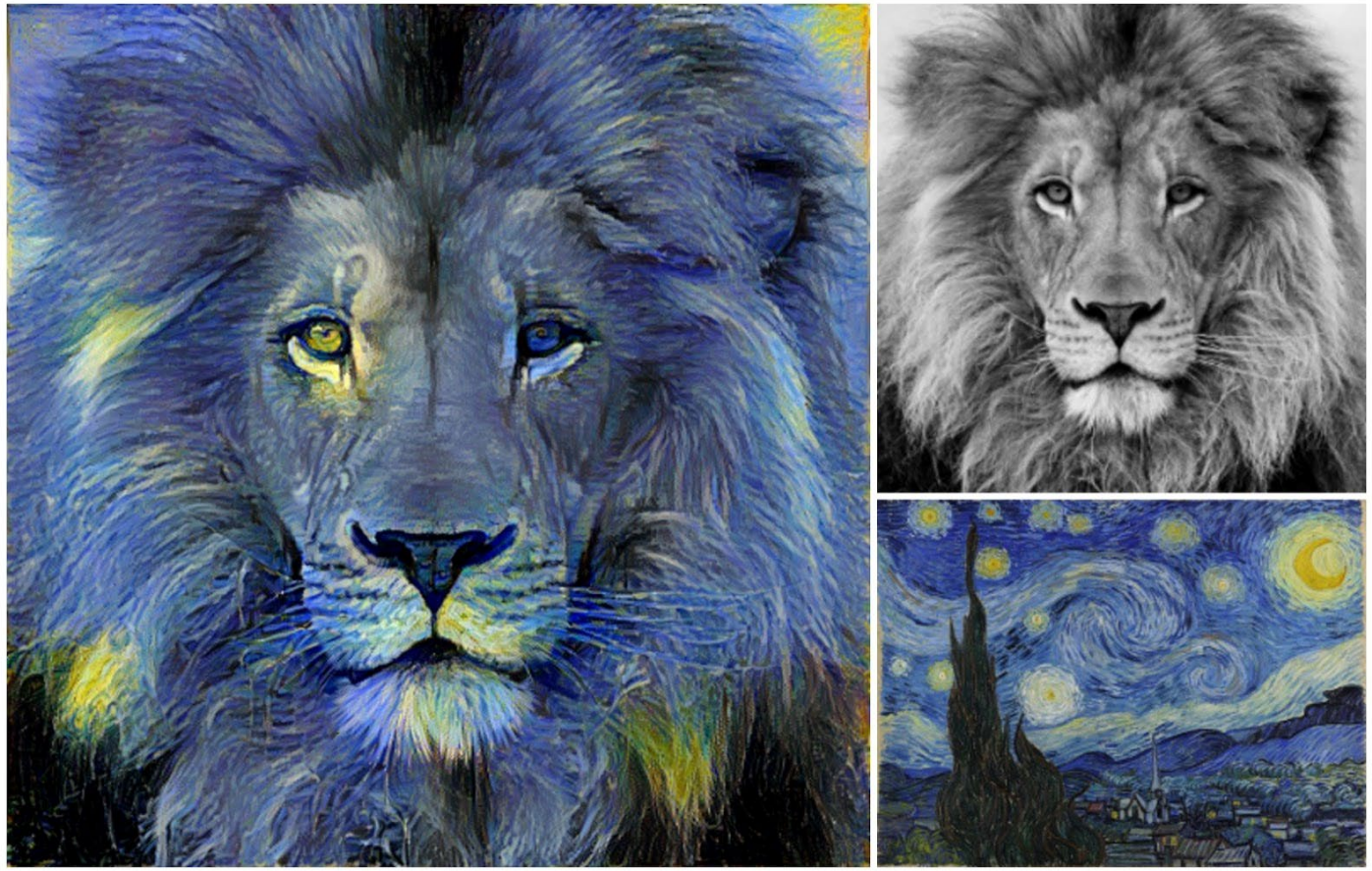


Image Style Transfer Using Convolutional Neural Networks (Gatys et al., 2016) Tensorflow adaptation by Cameroon Smith (cysmith@github)

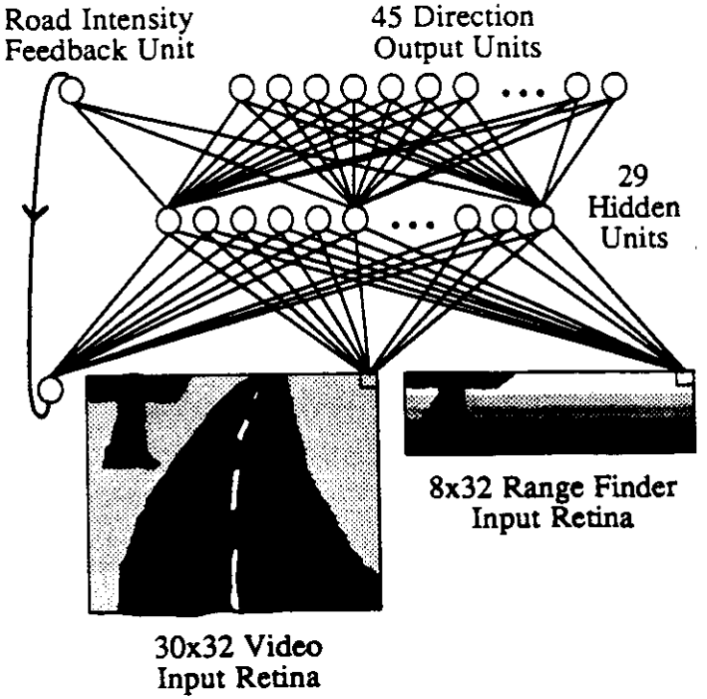
Short History

A short history of Neural Networks

- 1957: Perceptron (Frank Rosenblatt): one layer network neural network
- 1959: first neural network to solve a real world problem, i.e., eliminates echoes on phone lines (Widrow & Hoff)
- **First AI Winter**
- 1988: Backpropagation (Rumelhart, Hinton, Williams): learning a multi-layered network
- **Second AI Winter**

A short history of NNs

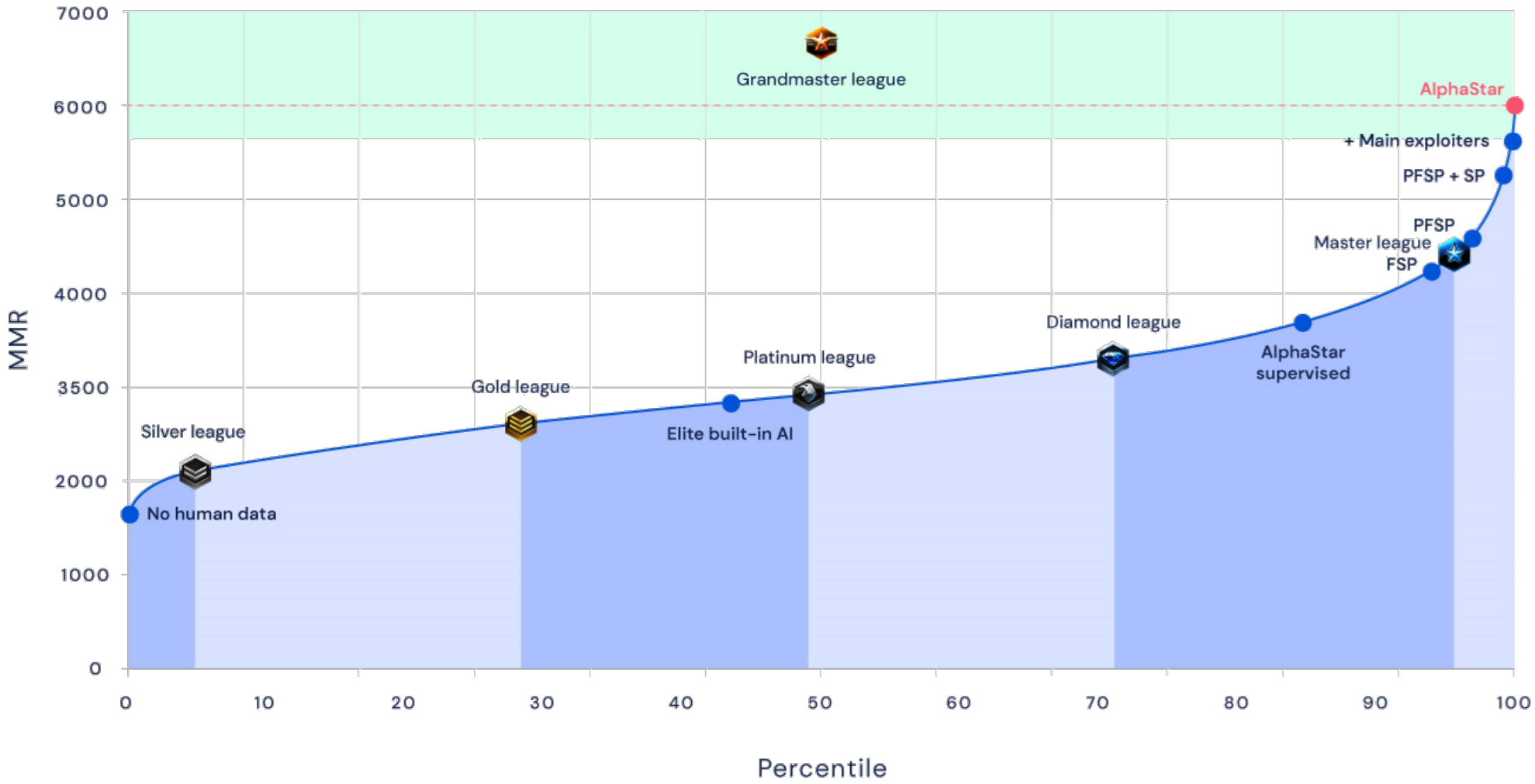
- 1989: ALVINN: autonomous driving car using NN (CMU)



A short history of NNs

- 1989: (LeCun) Successful application to recognize handwritten ZIP codes on mail using a “deep” network
- 2010s: near-human capabilities for image recognition, speech recognition, and language translation
- 2019 (AlphaStar) Google’s StarCraft 2 AI better than 99.8% of human players (GrandMaster level)

A short history of NNs



UNIVERSITY OF CALGARY

Basics

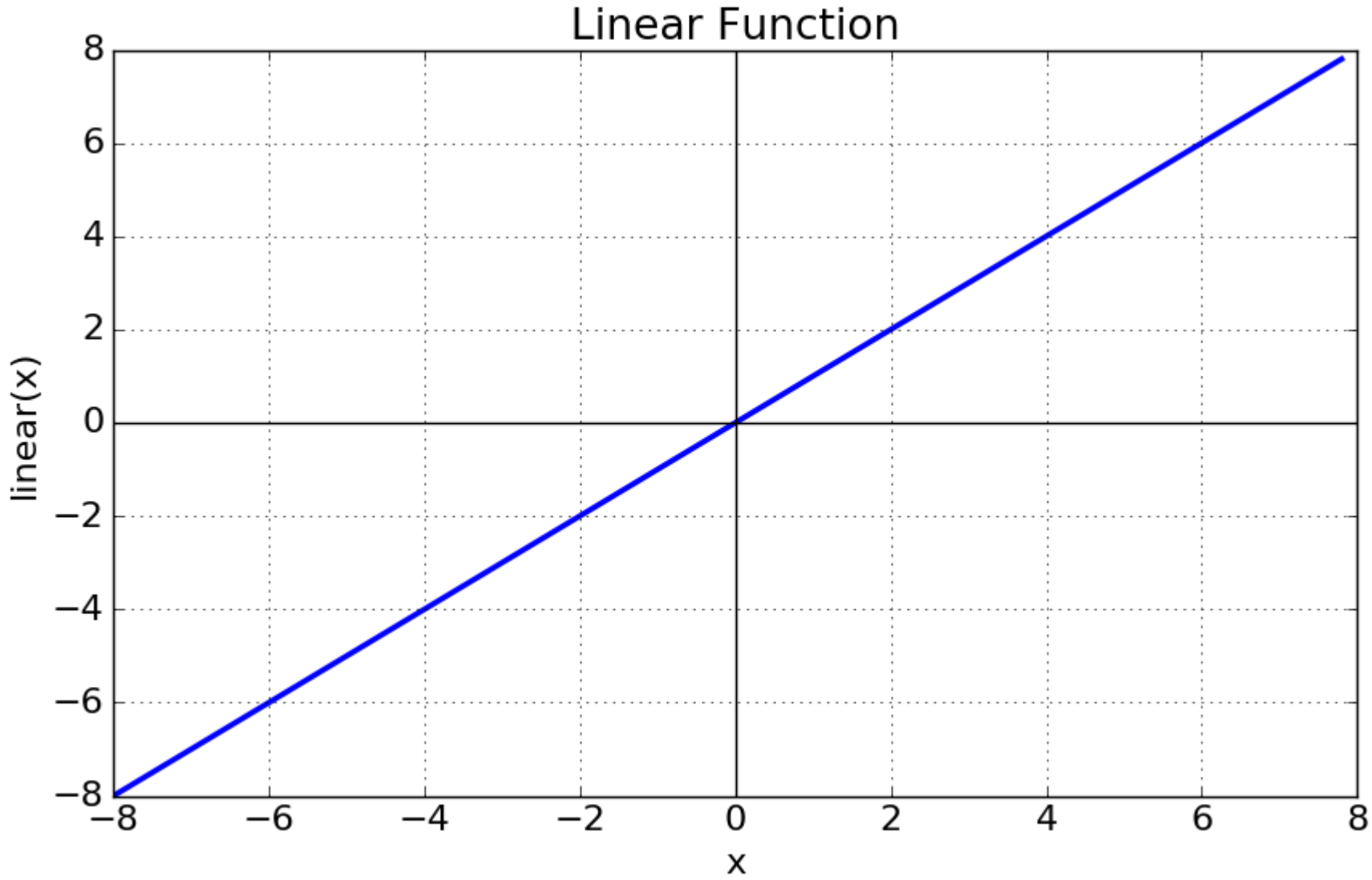
Basic data structures (I)

Directed, weighted graph:

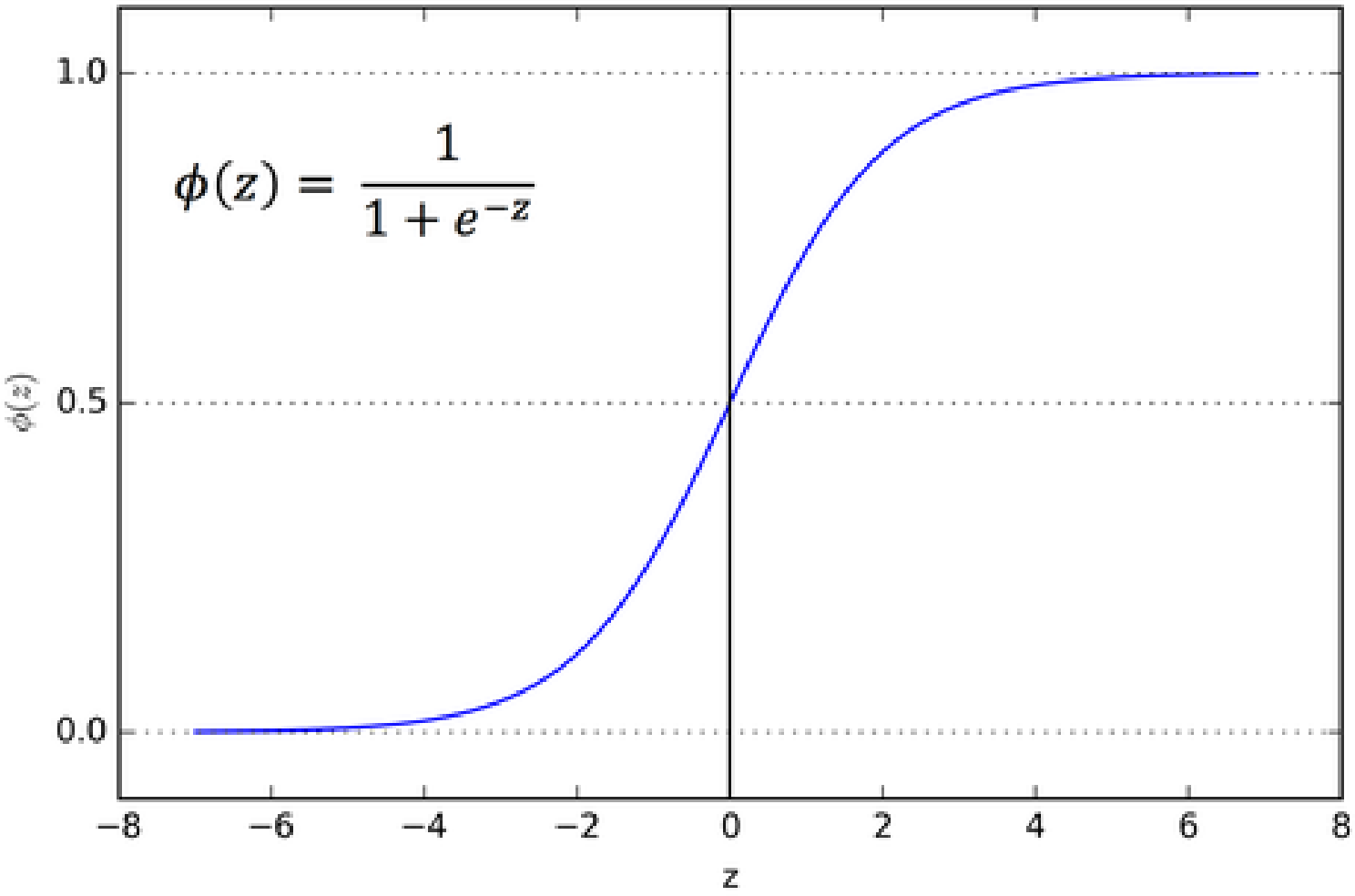
- Nodes represent a function (activation function f_{act}) with n arguments, if n links lead into the node, producing one result
  - Input nodes: take values from outside
  - Output nodes: represent activation values for different concepts to detect
  - Inner nodes: usually organized in layers (hidden layers)
- Labeled weighted links

Activation Functions

Activation Functions (Identity)

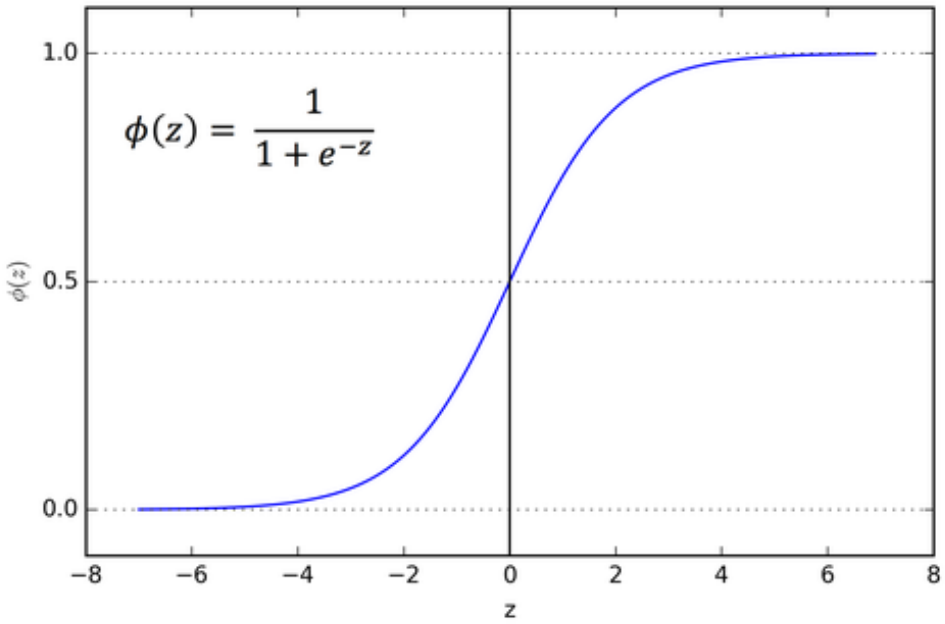


Sigmoid Activation Function



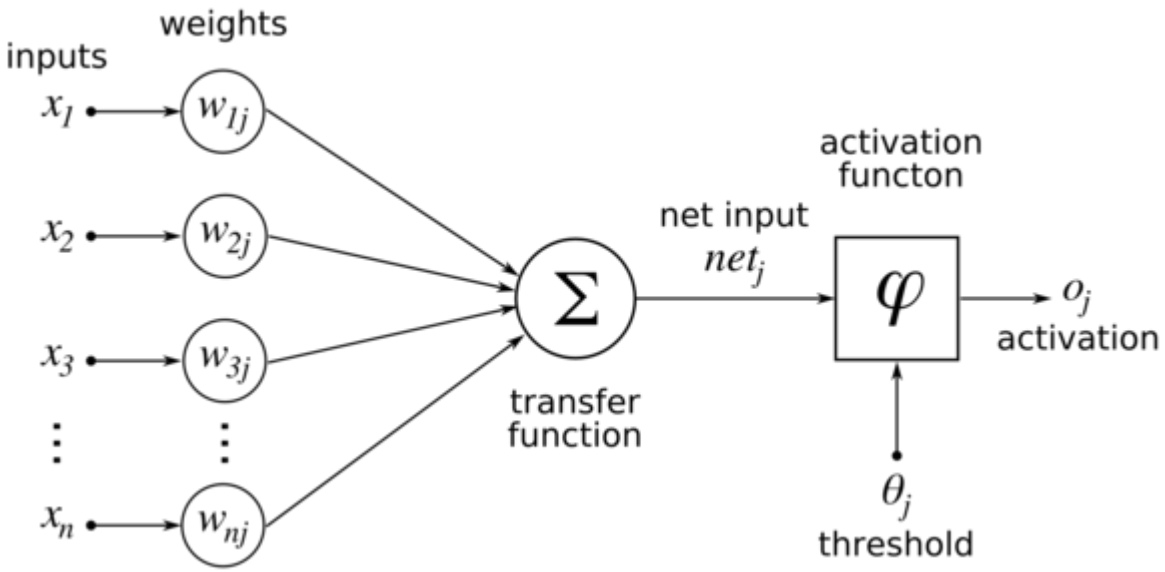
Sigmoid Activation Function

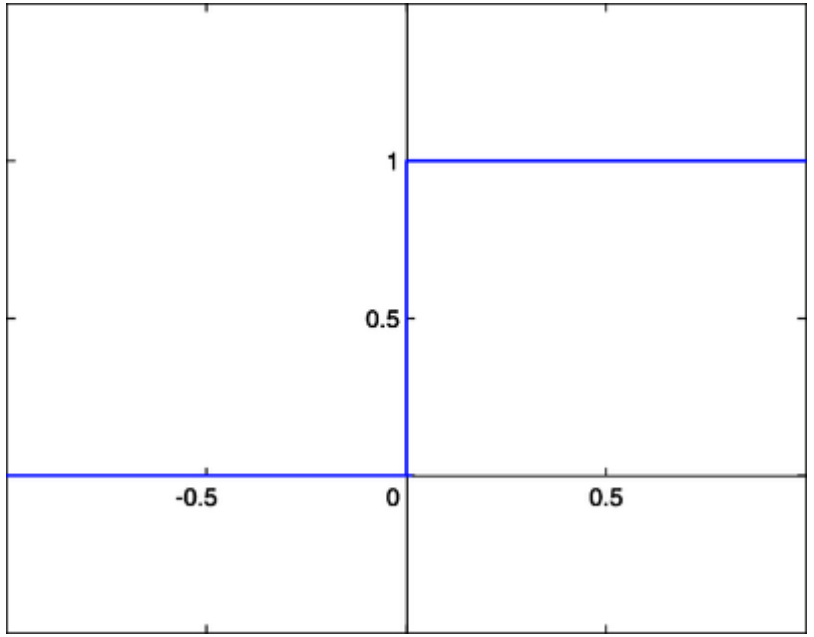
- The main reason why we use sigmoid function is because gradations exist between **(0 to 1)**
- Used for models where we have to **predict the probability** as an output.



Step (Sigmoid) Activation Function

- Replace a smooth function with a critical threshold point (if value exceeds then fully one answer)

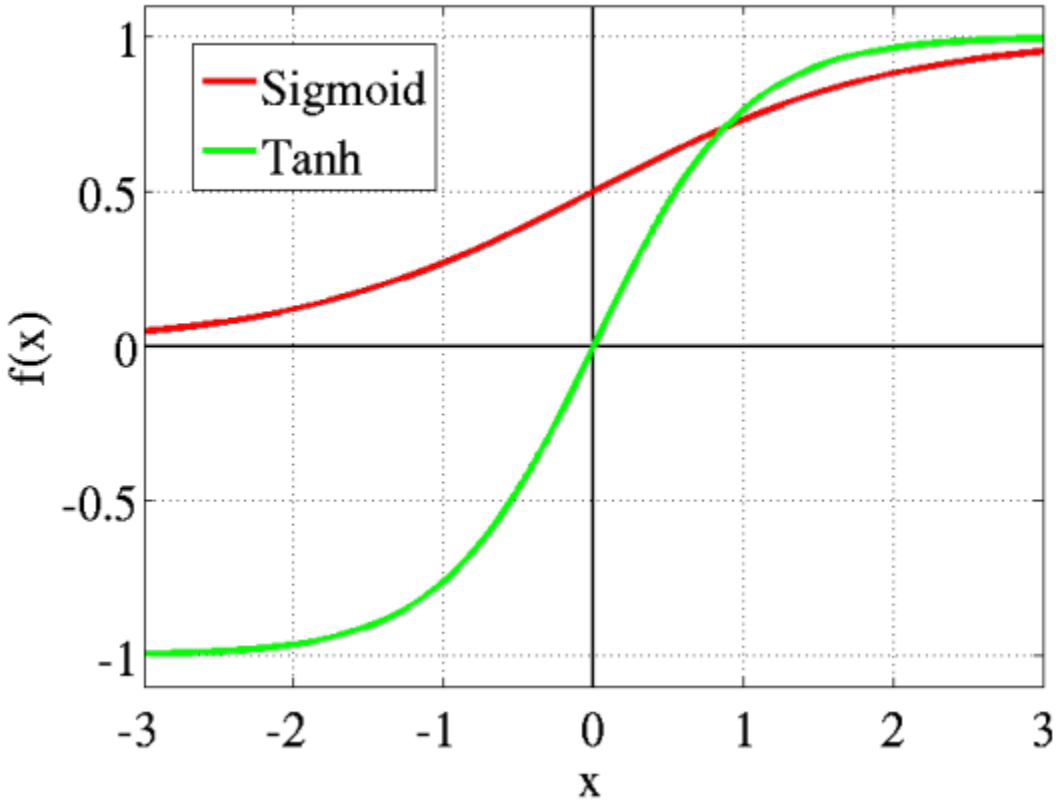




UNIVERSITY OF CALGARY

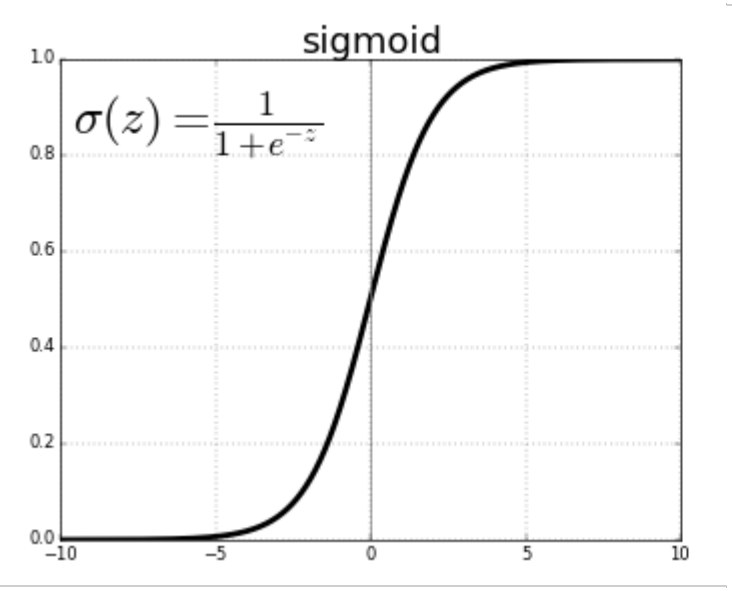
Tanh Activation Function

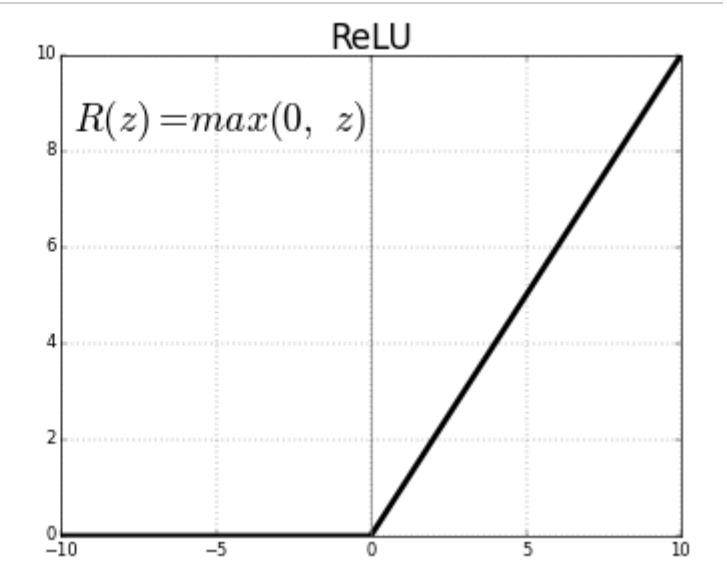
- The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.
- The tanh function is mainly used classification between two classes.



ReLU (Rectified Linear Unit) Activation Function

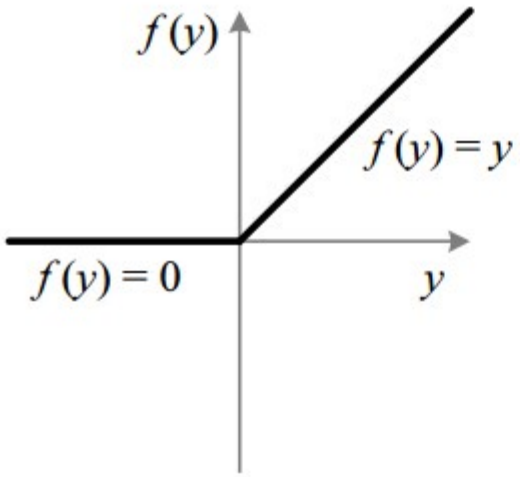
- The ReLU is the most used activation function in the world right now.
- Most used, due to Convolution Neural Network popularity
- But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly.

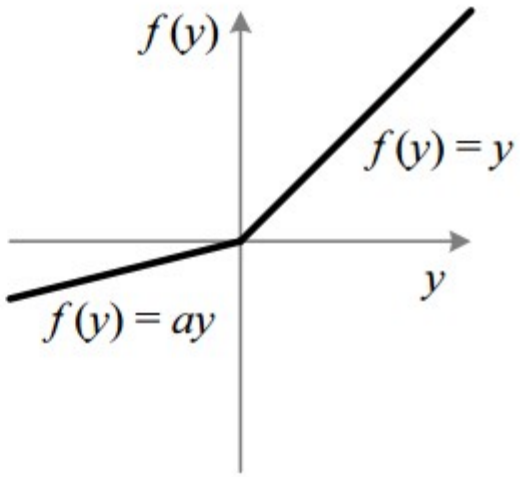




Leaky ReLU Activation Function

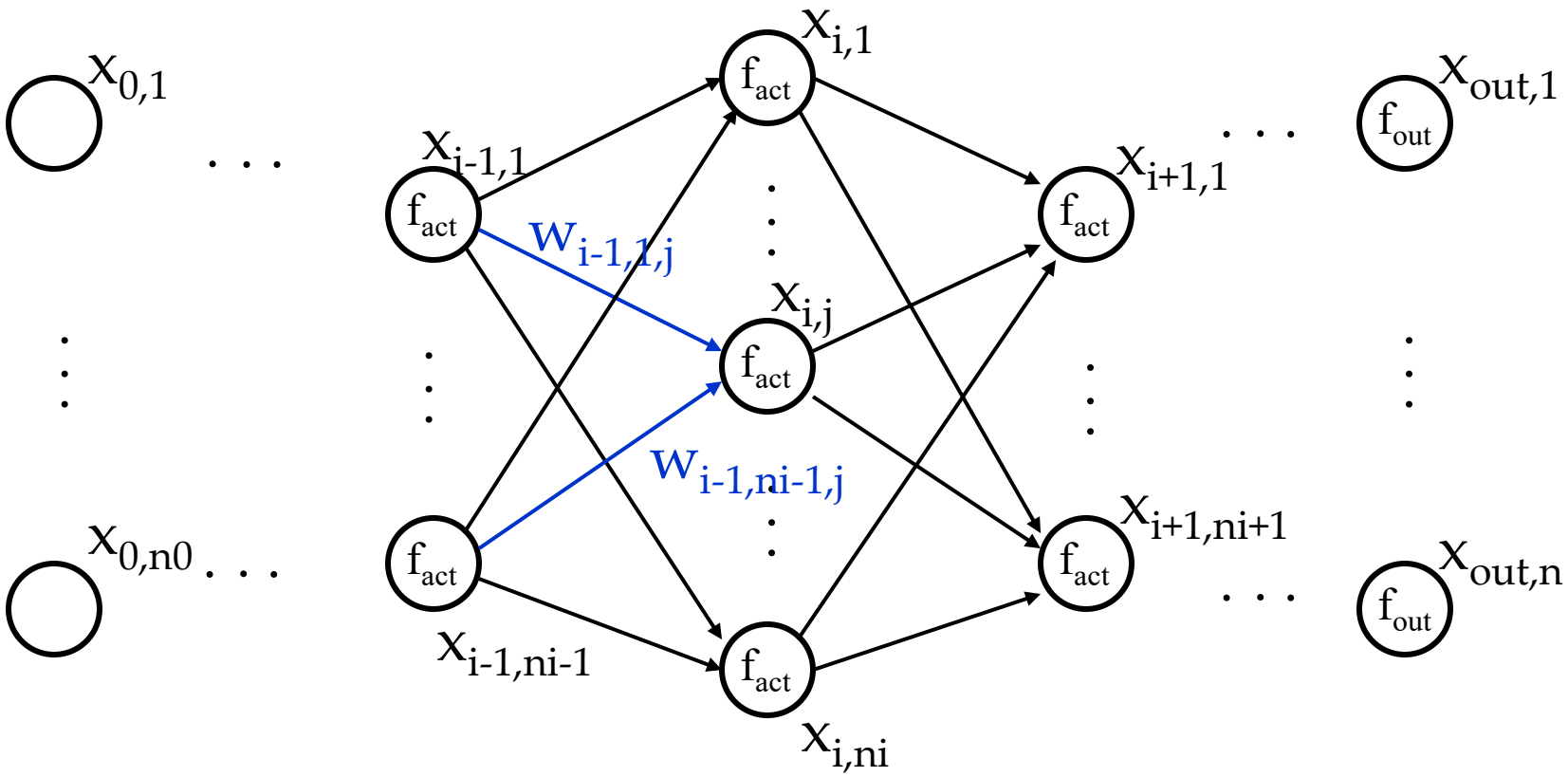
- It is an attempt to solve the dying ReLU problem
- The leak helps to increase the range of the ReLU function. Usually, the value of **a** is 0.01 or so.





Layers

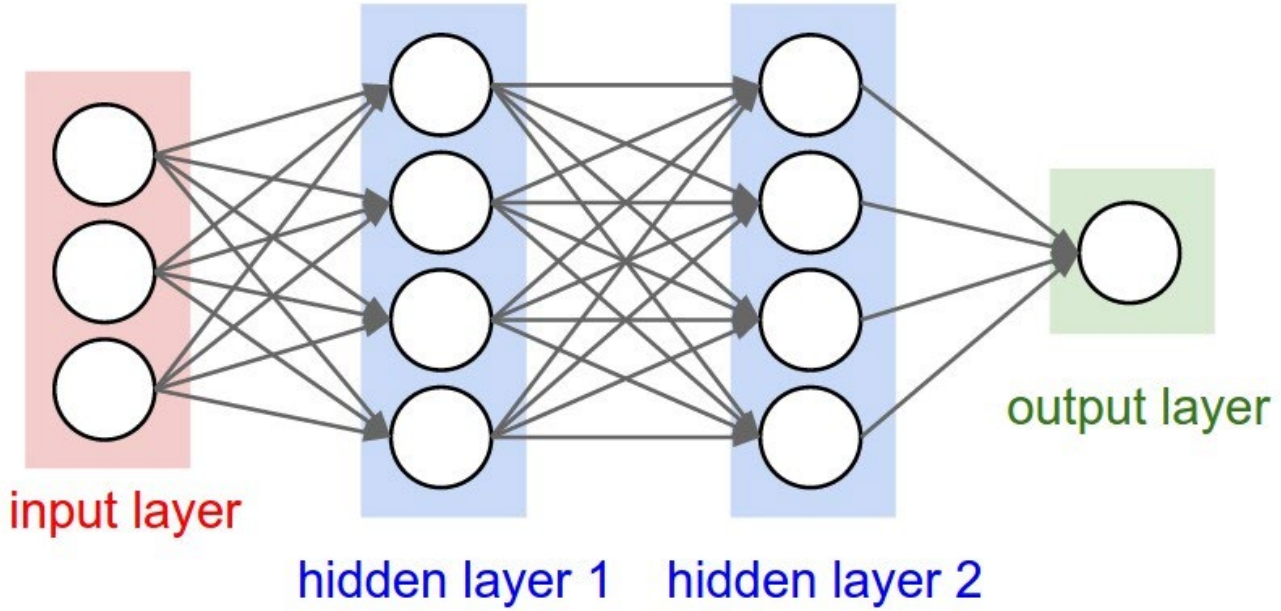
Basic data structures (II)



$$x_{ij} = f_{act}(x_{i-1,1} * w_{i-1,1,j}, \ldots, x_{i-1,ni-1} * w_{i-1,ni-1,j})$$

Hidden layers

- Hidden layers can find **features** within the data and allow following layers to operate on those features
  - Can implement XOR



From: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

Semantics

- Whole net represents a decision function linking input nodes to output nodes
- No other connection to knowledge or application area

How to get knowledge into the representation structure

By training the net!

General ideas:

- Provide set of input-output pattern
- Compare net behavior to expected output
- Adjust link weights according to result of comparison until error is minimized
  ☞ search problem

☞Learning process

Learning methods

General structure:

$w_{ijk,new} = f_{learn}(w_{ijk,old}, f_{act}, x_{ij}, x_{i+1,k}, \Delta x_{i+1,1},\ldots,\Delta x_{i+1,ni+1}, c)$

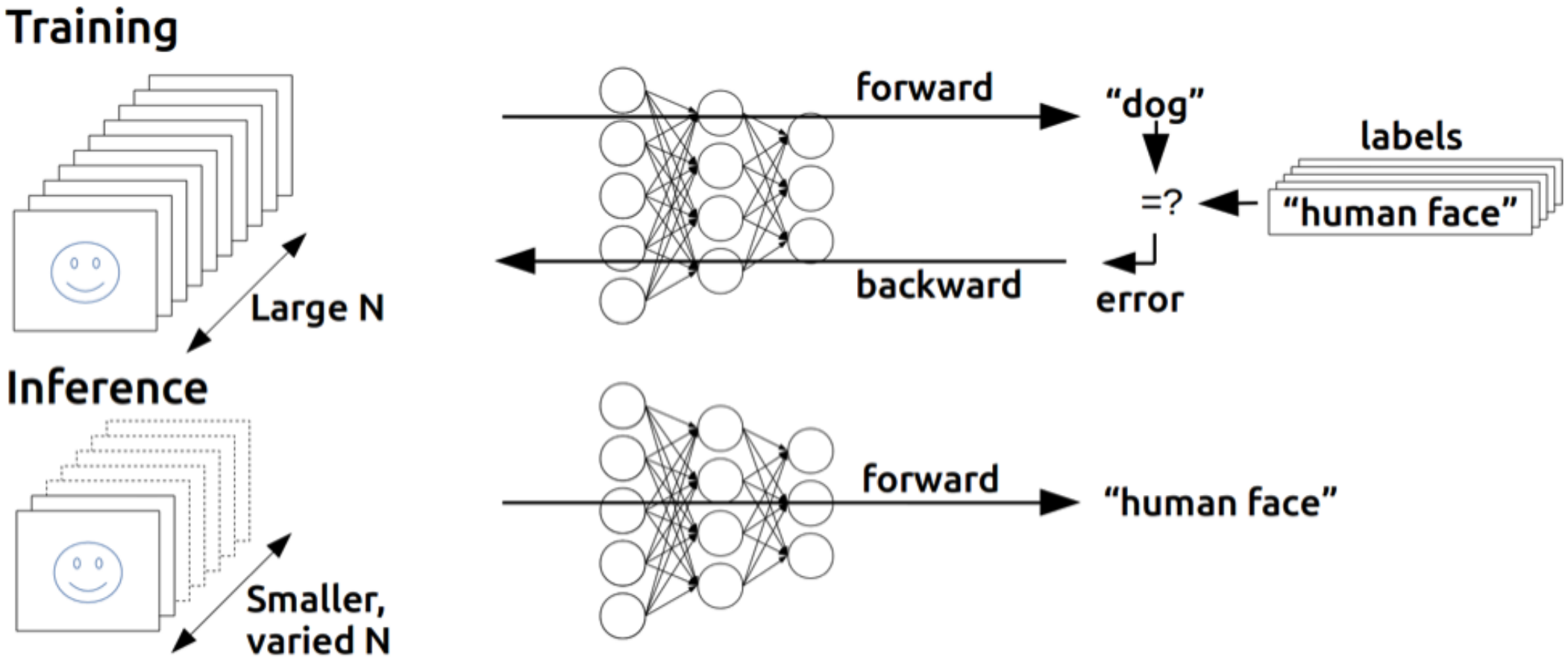
c learning factor (may change over time)

Examples:

- Perceptron training
- Backpropagation
- Winner takes all / Kohonen learning
- Counterpropagation
- …

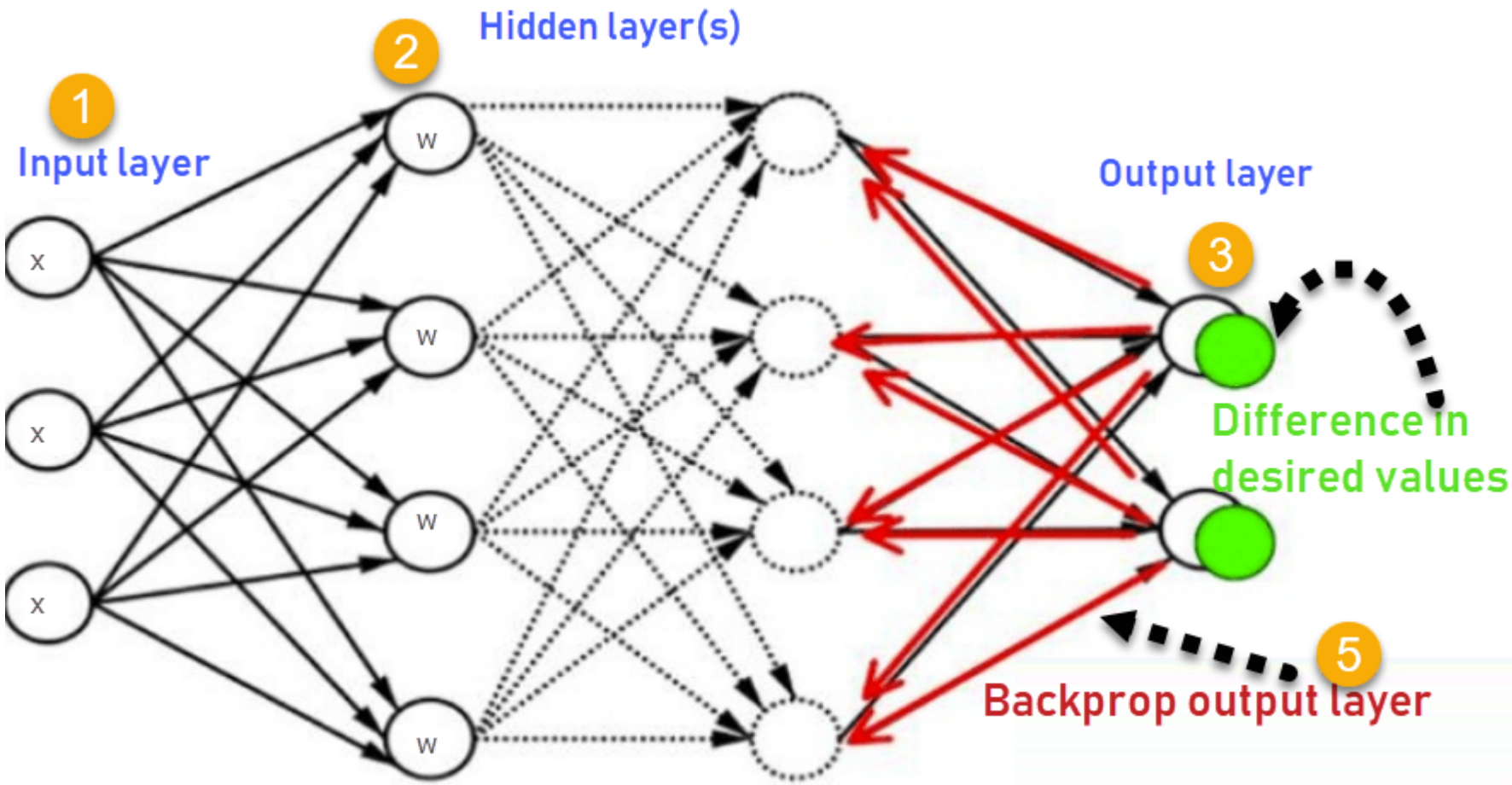
Backpropagation

Learning: Backpropagation

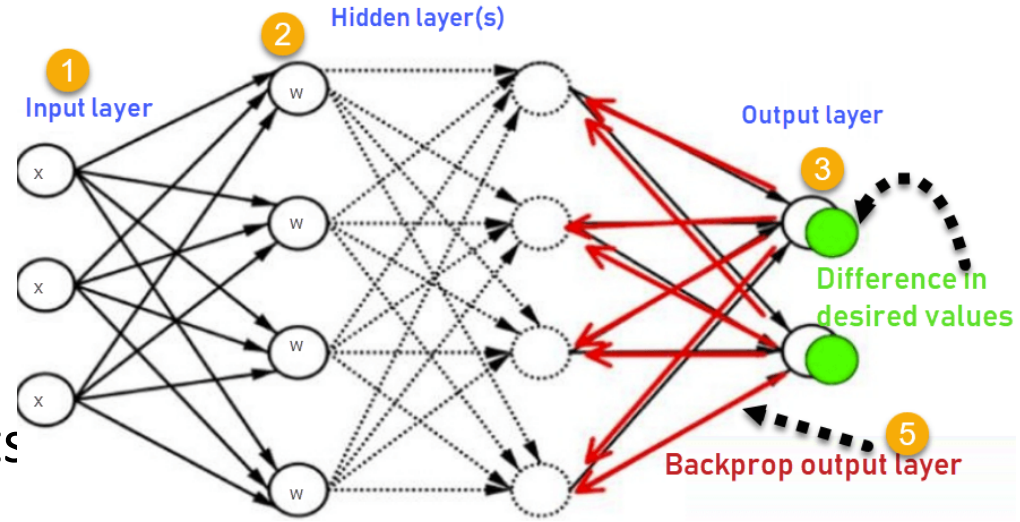


From: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

Back-Propagation



Back-Propagation



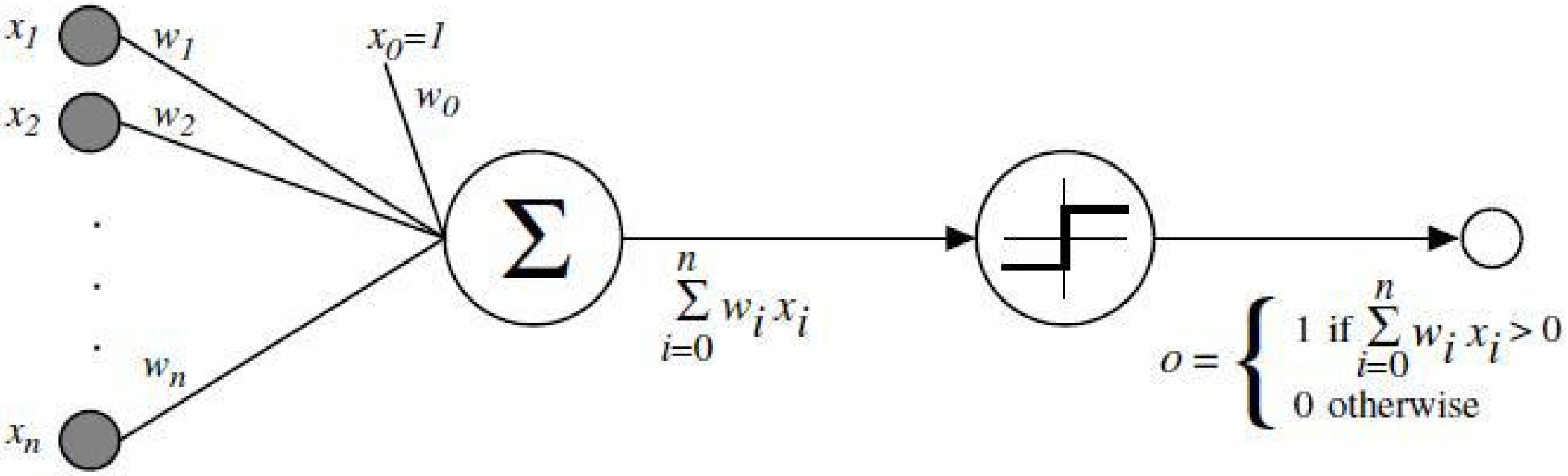
- Inputs X, arrive through the preconnected path
- Input is modeled using real weights W. The weights selected.
- Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
- Calculate the error in the outputs
- ErrorB= Actual Output – Desired Output
- Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.

Perceptron

Not a Dr. Who Robot (or maybe it is?)

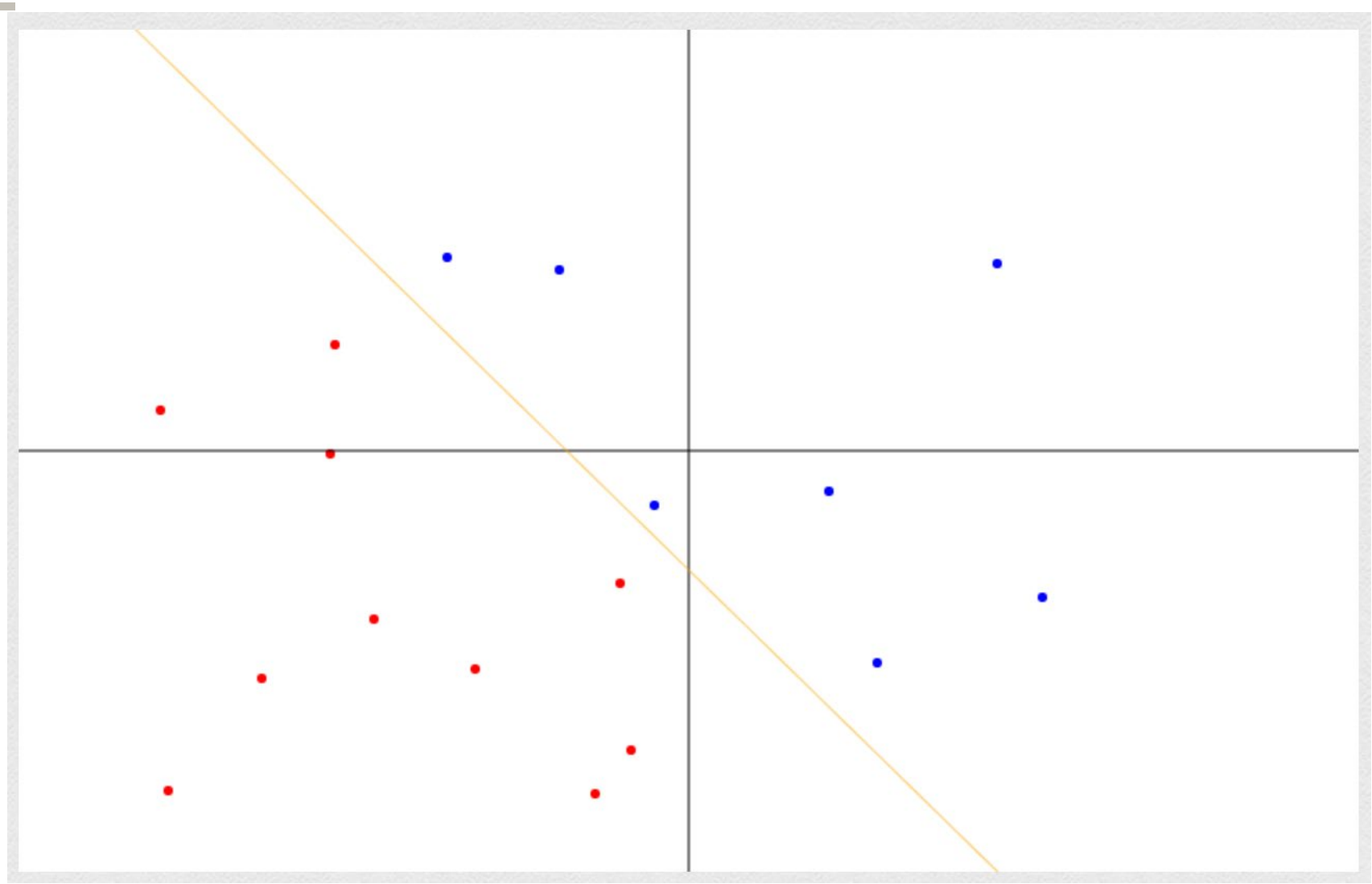
Perceptron

- Invented by Frank Rosenblatt (1957): simplified mathematical model of how the neurons in our brains operate



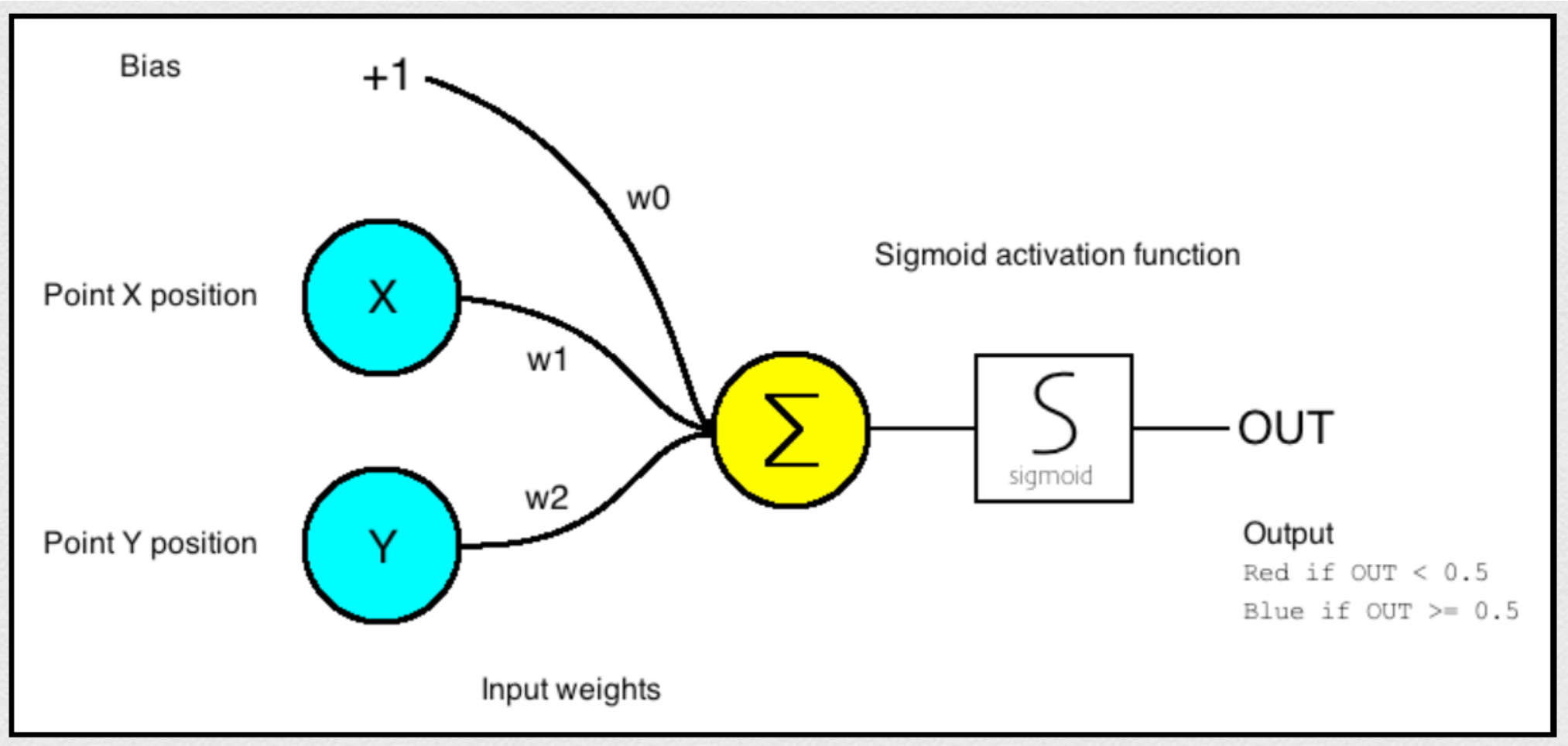
From: http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning/

Perceptron

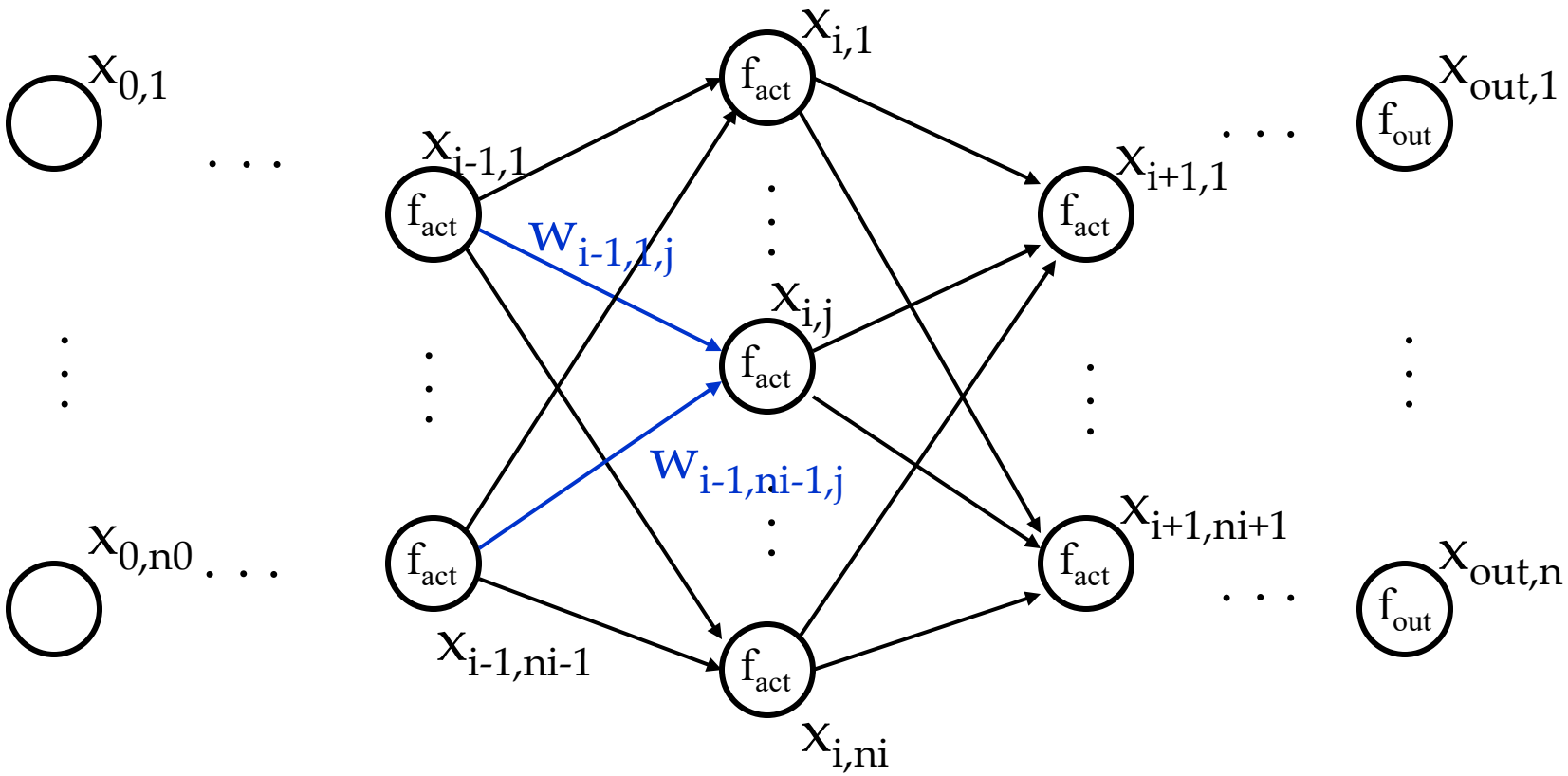


https://www.cs.utexas.edu/~teammco/misc/perceptron/

Perceptron



Basic data structures (II)



$$x_{ij} = f_{act}(x_{i-1,1} * w_{i-1,1,j}, \ldots, x_{i-1,ni-1} * w_{i-1,ni-1,j})$$

Example: Perceptron (I)

- No hidden layers
- $f_{act}\ (a_1,\ldots,a_m) = sig(a_1+\ldots+a_m)$
- $f_{learn}(w_{0j1,old},d,x_{01},\ldots,x_{0k},c) =$
  $w_{0j1,old} + c*(d - sig(\sum x_{0i}*w_{0i1}))*x_{0j}$
  where d is the expected output of the web
- Usually there are more inputs to a perceptron than what is suggested by the function to learn:
  bias-nodes allow for more learning accuracy

Example: Perceptron (II)

Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

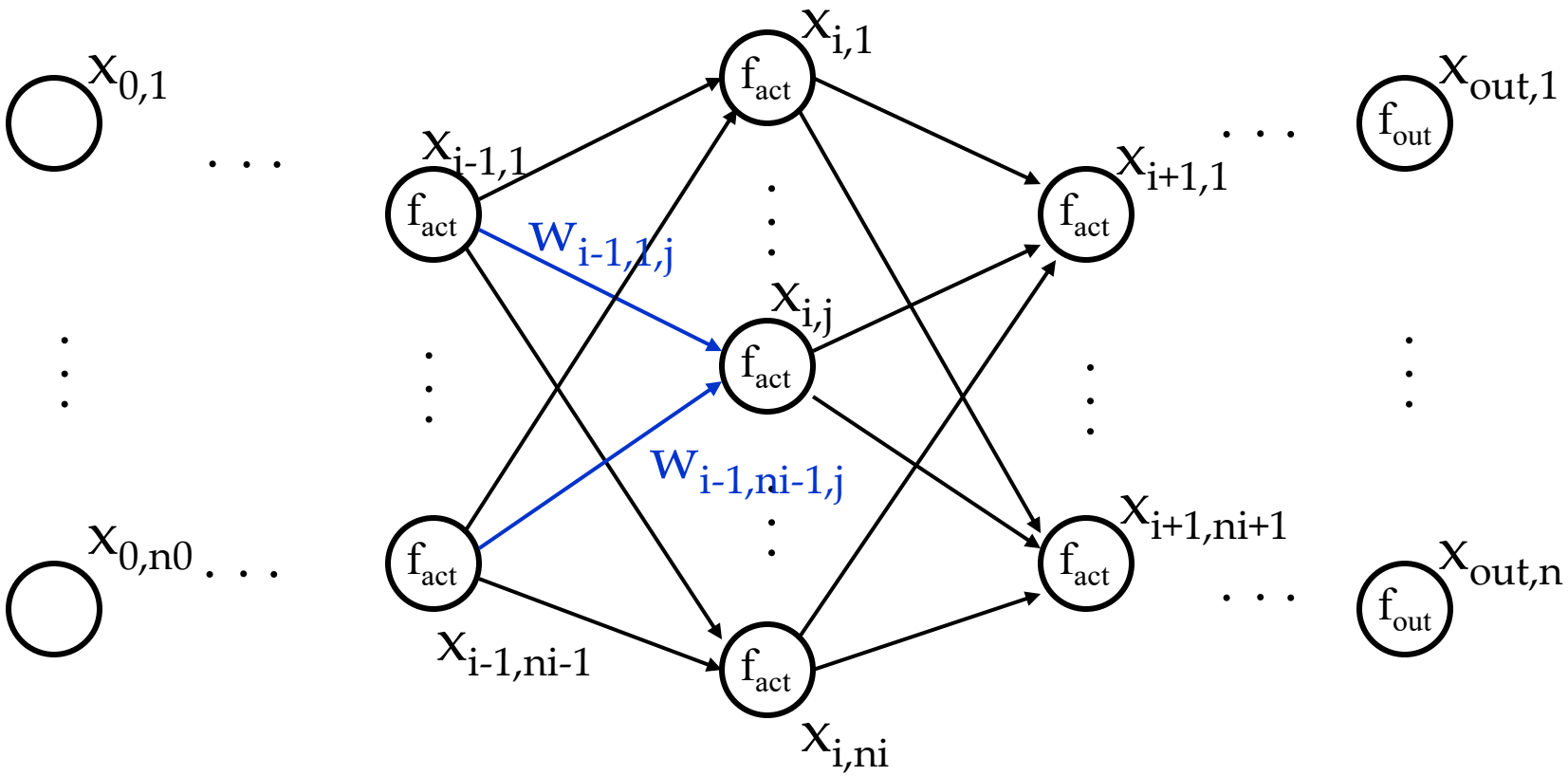
Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

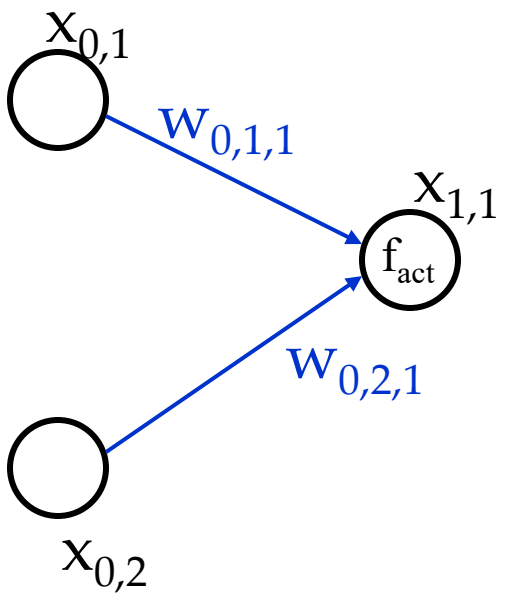
(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1

Basic data structures (II)



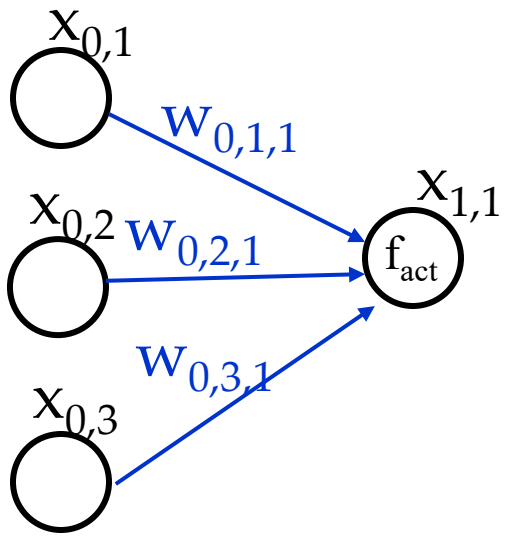
$$x_{ij} = f_{act}(x_{i-1,1} * w_{i-1,1,j}, \ldots, x_{i-1,ni-1} * w_{i-1,ni-1,j})$$

Basic data structures (II)



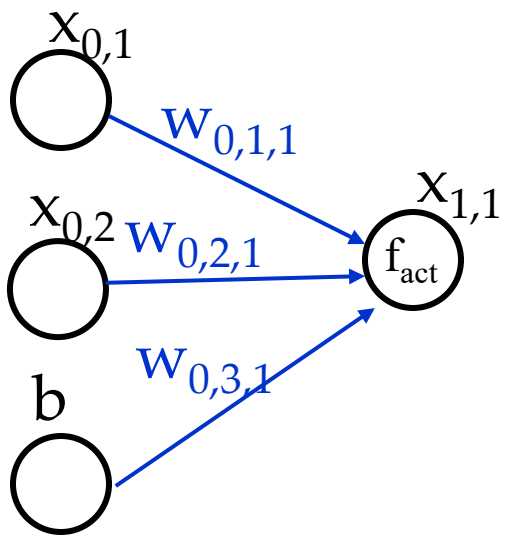
$$x_{11} = f_{act}(x_{0,1} * w_{0,1,1}, x_{0,2} * w_{0,2,1})$$

Basic data structures (II)



$$x_{11} = f_{act}(x_{0,1}*w_{0,1,1}, x_{0,2}*w_{0,2,1}, x_{0,3}*w_{0,3,1})$$

Basic data structures (II)



$$x_{11} = f_{act}(x_{0,1}*w_{0,1,1}, x_{0,2}*w_{0,2,1}, b*w_{0,3,1})$$

Example: Perceptron (II)

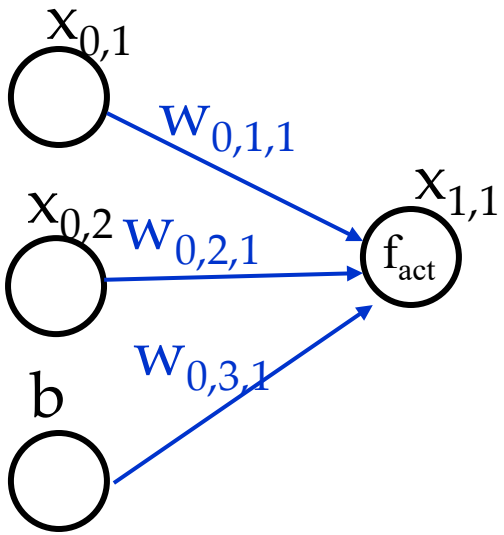
Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1



Example: Perceptron (II)

Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

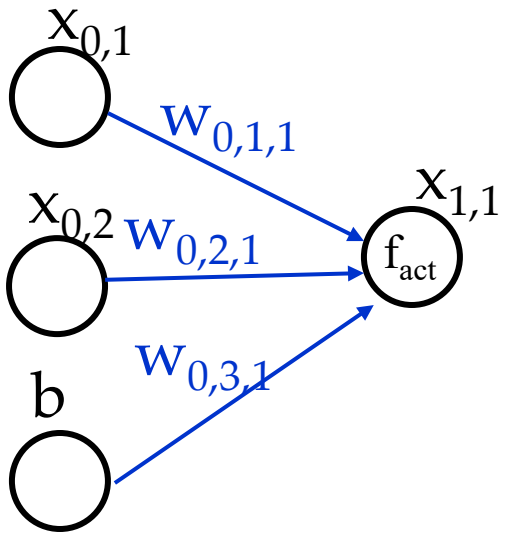
Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1

- $f_{learn}(w_{0j1,old}, d, x_{01}, \ldots, x_{0k}, c) = w_{0j1,old} + c*(d - sig(\sum x_{0i} * w_{0i1})) * x_{0j}$



Example: Perceptron (II)

Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

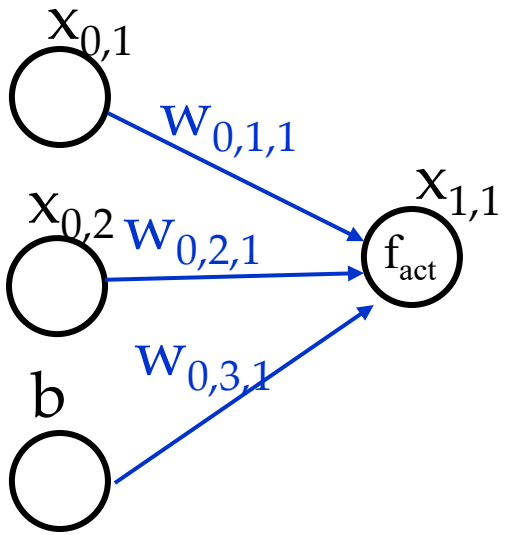
Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1

- $w_{011,new} = w_{011,old} + c*(d - sig(\sum x_{0i}*w_{0i1}))*x_{01}$
- $w_{021,new} = w_{021,old} + c*(d - sig(\sum x_{0i}*w_{0i1}))*x_{02}$
- $w_{031,new} = w_{031,old} + c*(d - sig(\sum x_{0i}*w_{0i1}))*b$



Example: Perceptron (II)

Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

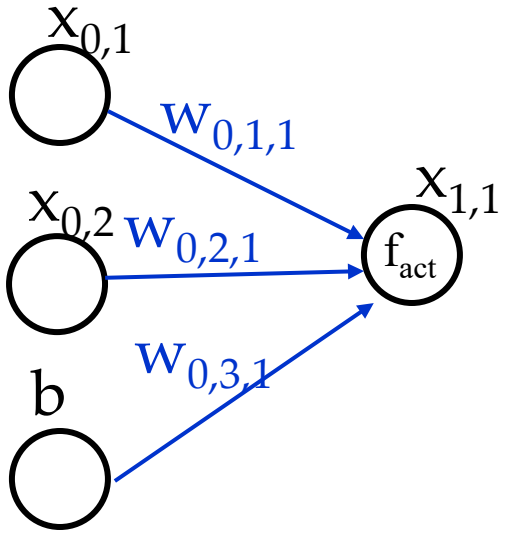
Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1

- $w_{011,new} = 0.3 + 0.3 * (1 - sig(\sum x_{0i} * w_{0i1})) * 1$
- $w_{021,new} = 0.4 + 0.3 * (1 - sig(\sum x_{0i} * w_{0i1})) * 1$
- $w_{031,new} = -0.2 + 0.3 * (1 - sig(\sum x_{0i} * w_{0i1})) * 1$



Example: Perceptron (II)

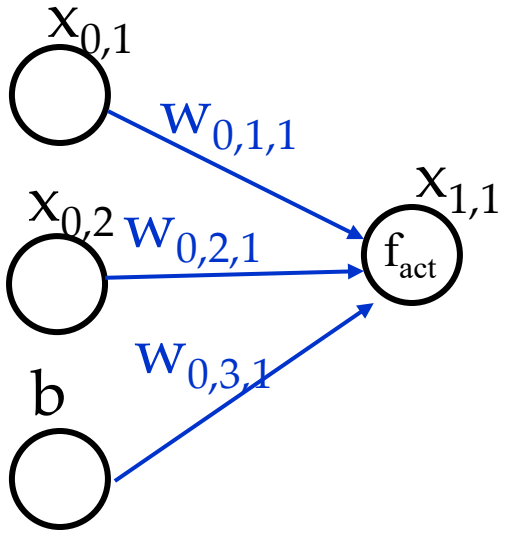
Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1



- $w_{011,new} = 0.3 + 0.3 *(1 - sig(\sum x_{0i} * w_{0i1})) * 1$
- $w_{021,new} = 0.4 + 0.3 *(1 - sig(\sum x_{0i} * w_{0i1})) * 1$
- $w_{031,new} = -0.2 + 0.3 *(1 - sig(\sum x_{0i} * w_{0i1})) * 1$
- $sig(\sum x_{0i} * w_{0i1}) = sig(0.3*1 + 0.4*1 + -0.2*1) = sig(0.5) = 1$

Heaviside step function

Example: Perceptron (II)

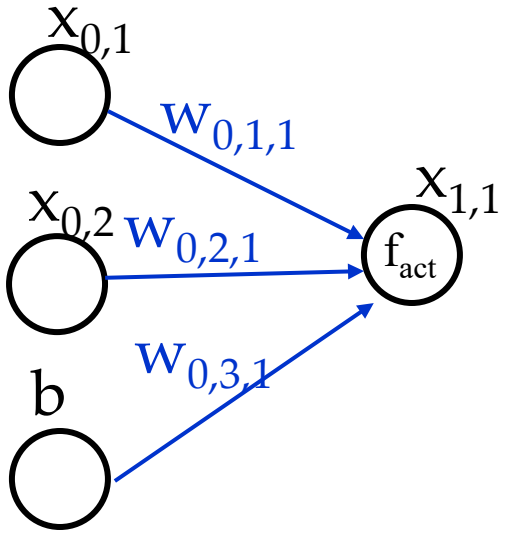
Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1



- $w_{011,new} = 0.3 + 0.3 * (1 - 1) * 1 = 0.3$
- $w_{021,new} = 0.4 + 0.3 * (1 - 1) * 1 = 0.4$
- $w_{031,new} = -0.2 + 0.3 * (1 - 1) * 1 = -0.2$
- $sig(\sum x_{0i} * w_{0i1}) = sig(0.3*1 + 0.4*1 + -0.2*1) = sig(0.5) = 1$

Heaviside step function

Example: Perceptron (II)

Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

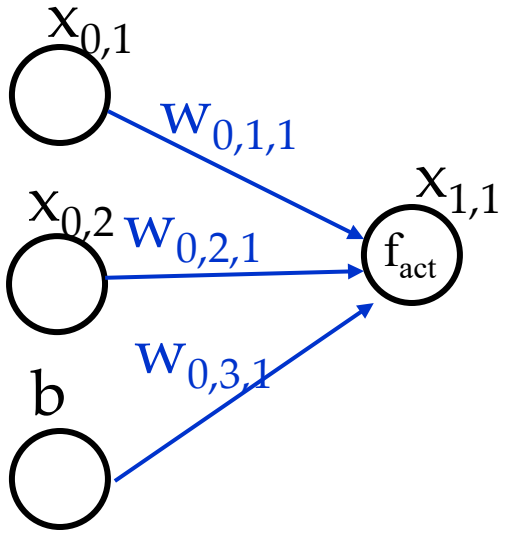
Let the initial weights be

$w_{011} = 0.3$; $w_{021} = 0.4$; $w_{031} = -0.2$

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1

- $w_{011,new} = 0.3 + 0.3 * (-1 - 1) * 9.4 = -5.34$
- $w_{021,new} = 0.4 + 0.3 * (-1 - 1) * 6.4 = -3.44$
- $w_{031,new} = -0.2 + 0.3 * (-1 - 1) * 1 = -0.8$
- $sig(\sum x_{0i} * w_{0i1}) = sig(0.3*9.4 + 0.4*6.4 + -0.2*1) = sig(5.18) = 1$



Restart!

Heaviside step function

Example: Perceptron (II)

Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

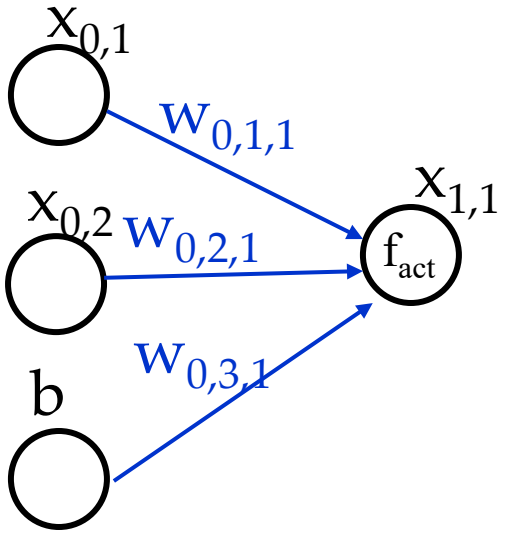
Let the initial weights be

w_{011} = -5.34 ; w_{021} = -3.44 ; w_{031} = -0.8

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1

- $w_{011,new}$ = -5.34 + 0.3 *(1 - -1)*1 = -4.74
- $w_{021,new}$ = -3.44 + 0.3 *(1 - -1)*1 = -2.84
- $w_{031,new}$ = -0.8 + 0.3 *(1 - -1)*1 = -0.2
- sig($\sum x_{0i} * w_{0i1}$) = sig(-5.34*1 + -3.44*1 + -0.8*1) = sig(-9.58) = -1



Restart!

Heaviside step function

Example: Perceptron (Finally) (8th Try Through)

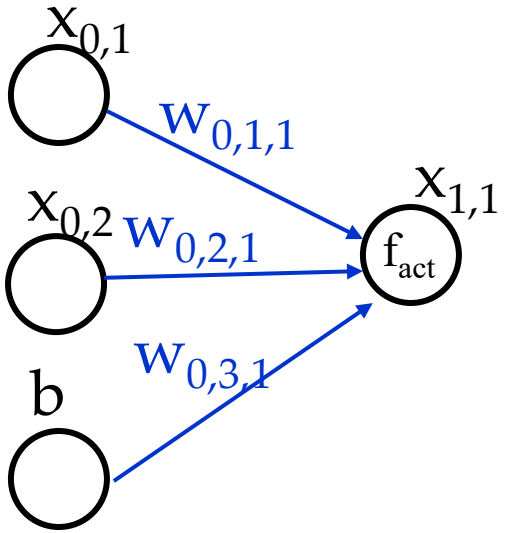
Perform the learning for a perceptron with two input nodes and one bias node with constant value of 1 and a learning rate c = 0.3

Let the initial weights be

w_{011} = -1.74 ; w_{021} = 0.16 ; w_{031} = 2.8

Training data:

(1,1) -> 1; (9.4,6.4) -> -1; (8,7.7) -> -1; (0.5,2.2) -> 1



- $w_{011,new}$ = -1.74 + 0.3 *(1 - 1)*0.5 = -4.74
- $w_{021,new}$ = 0.16 + 0.3 *(1 - 1)*2.2 = -2.84
- $w_{031,new}$ = 2.8 + 0.3 *(1 - 1)*1 = -0.2
- sig($\sum x_{0i} * w_{0i1}$) = sig(-1.74*0.5 + 0.16*2.2 + 2.8*1) = sig(2.282) = 1

Heaviside step function

Example: Perceptron (Finally) (8th Try Through)

- Series of w011, w021, w031 values

0.30	0.40	-0.2
-5.34	-3.44	-0.8
-4.74	-2.84	-0.2
-4.14	-2.24	0.4
-3.54	-1.64	1.0
-2.94	-1.04	1.6
-2.34	-0.44	2.2
-1.74	0.16	2.8

Lesson: ImageNet

Deep Learning/ Convolutional Neural Networks

- Classify an image into 1000 possible classes:
  e.g. Abyssinian cat, Bulldog, French Terrier, Cormorant, Chickadee, red fox, banj
  o, barbell, hourglass, knot, maze, viaduct, etc.

cat, tabby cat (0.71)
Egyptian cat (0.22)
red fox (0.11)
…..

The Data: ILSVRC

- Imagenet Large Scale Visual Recognition Challenge (ILSVRC): Annual Competition

1000 Categories

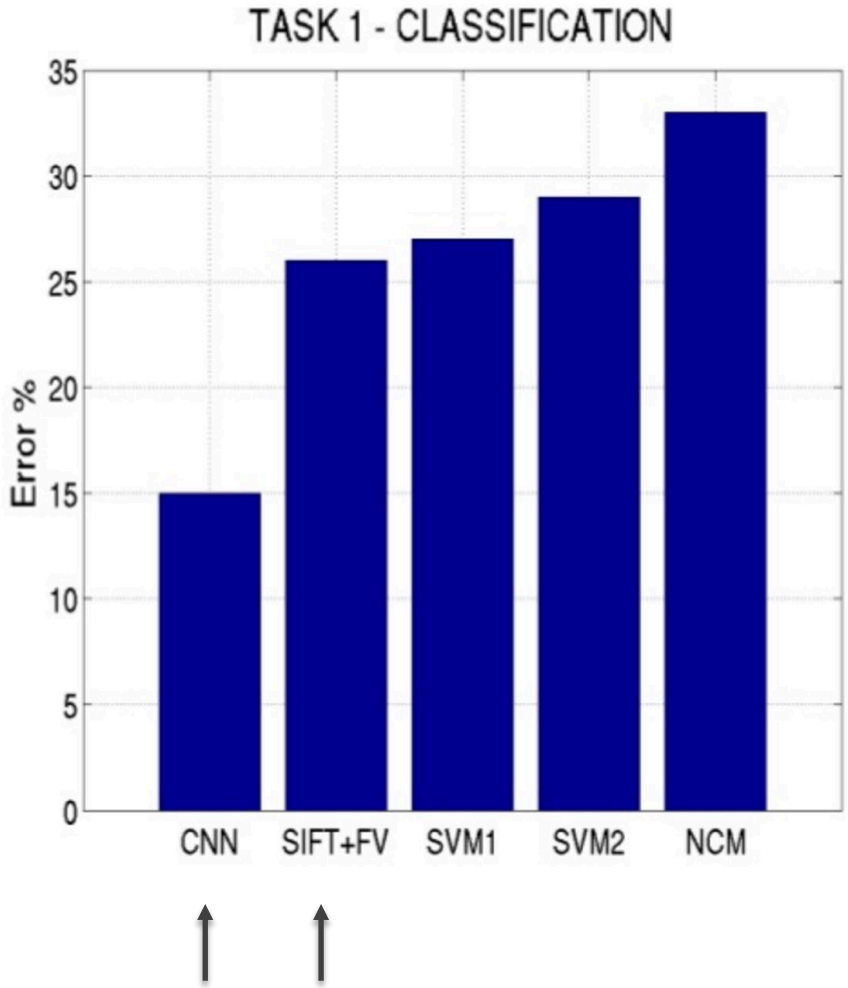
~1000 training images per Category

~1 million images in total for training

~50k images for validation

Only images released for the test set but no annotations,
evaluation is performed centrally by the organizers (max 2 per week)

Top-5 error on this competition (2012)



Context (circa 2015)

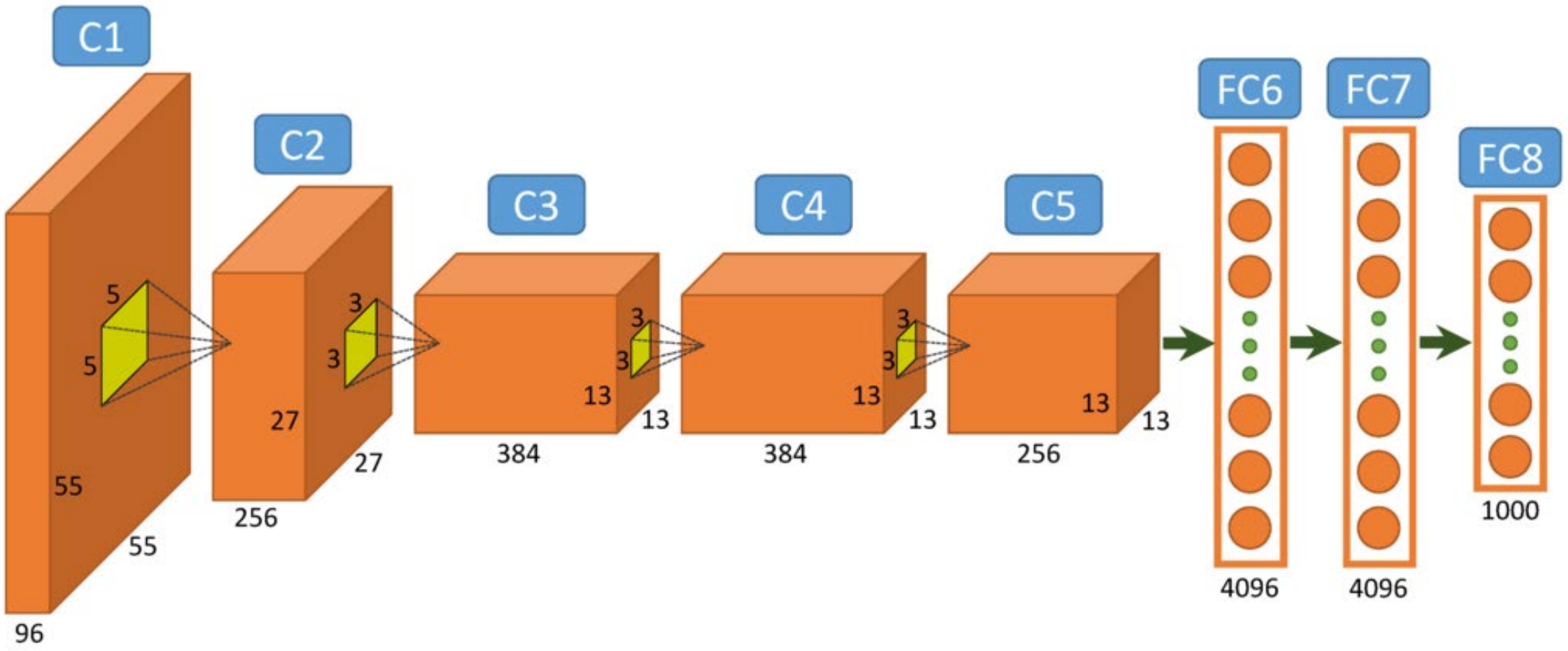
- Deep learning already claiming big successes

Team	Year	Place	Error (top-5)
XRCE (pre-neural-net explosion)	2011	1st	25.8%
Supervision (AlexNet)	2012	1st	16.4%
Clarifai	2013	1st	11.7%
GoogLeNet (Inception)	2014	1st	6.66%
Andrej Karpathy (human)	2014	N/A	5.1%
BN-Inception (Arxiv)	2015	N/A	4.9%
Inception-v3 (Arxiv)	2015	N/A	3.46%

Imagenet challenge classification task

From: http://www.wsdm-conference.org/2016/slides/WSDM2016-Jeff-Dean.pdfz

Alexnet



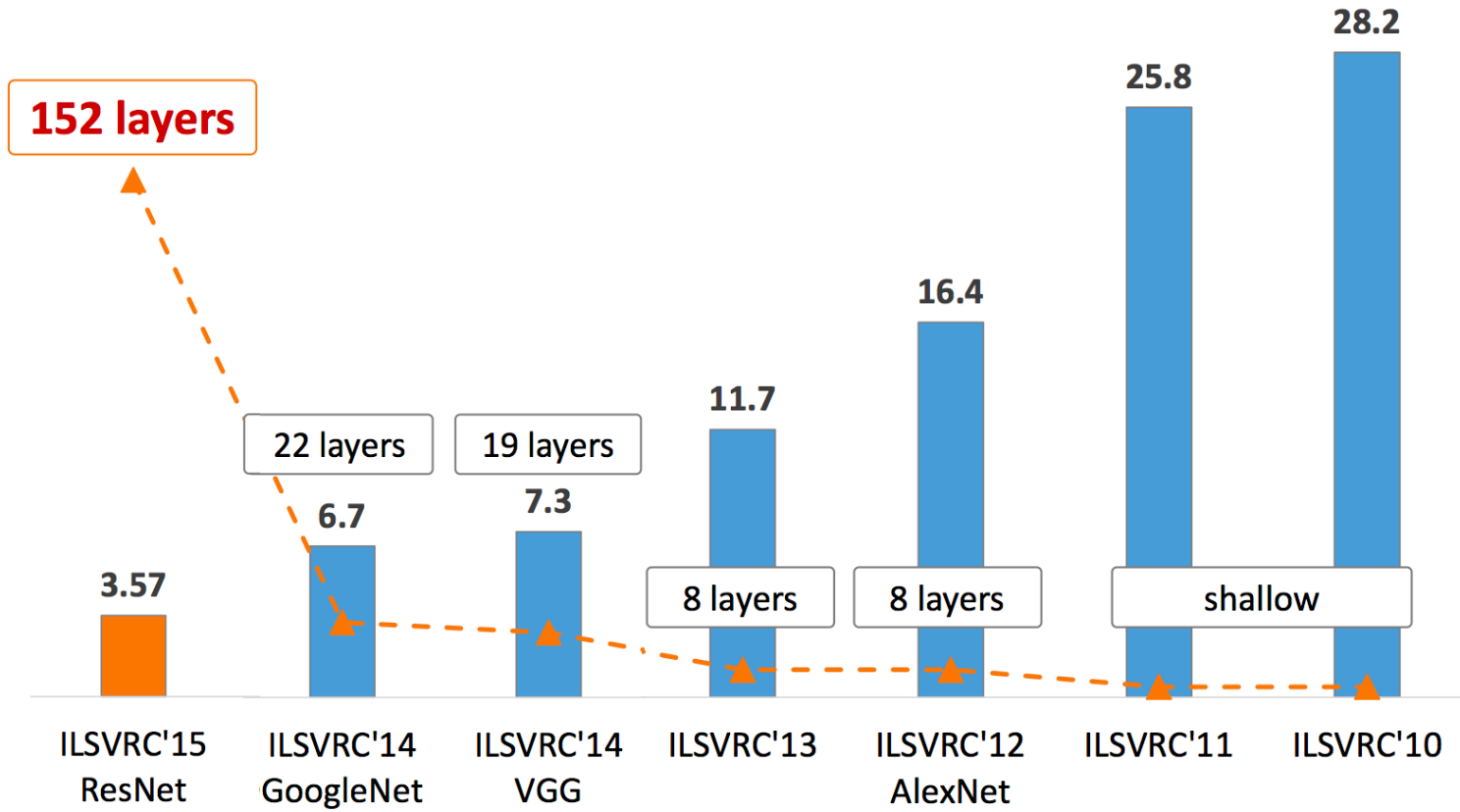
https://www.saagie.com/fr/blog/object-detection-part1

Revolution of Depth

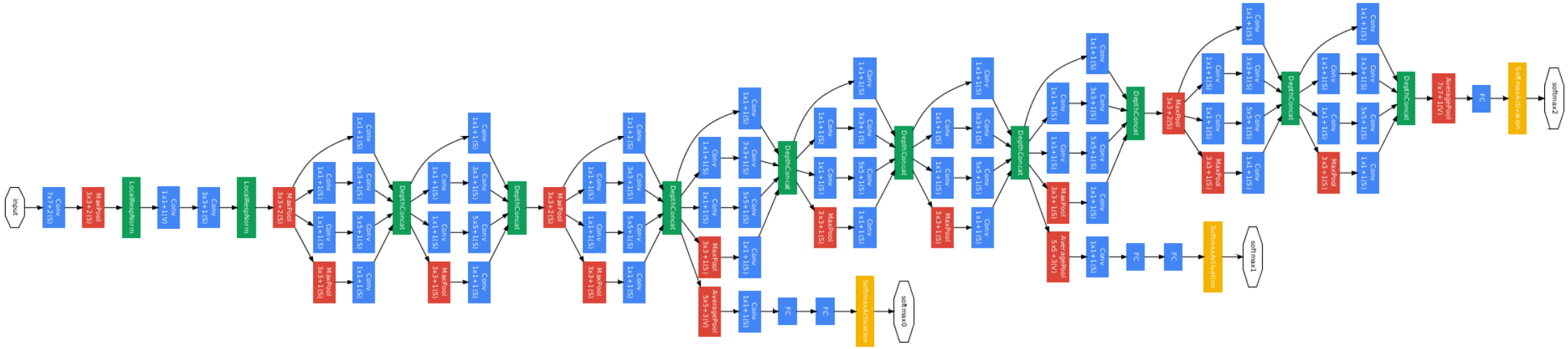
AlexNet, 8 layers
(ILSVRC 2012)

VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)



GoogLeNet



Keras: https://gist.github.com/joelouismarino/a2ede9ab3928f999575423b9887abd14

Szegedy et al. 2014

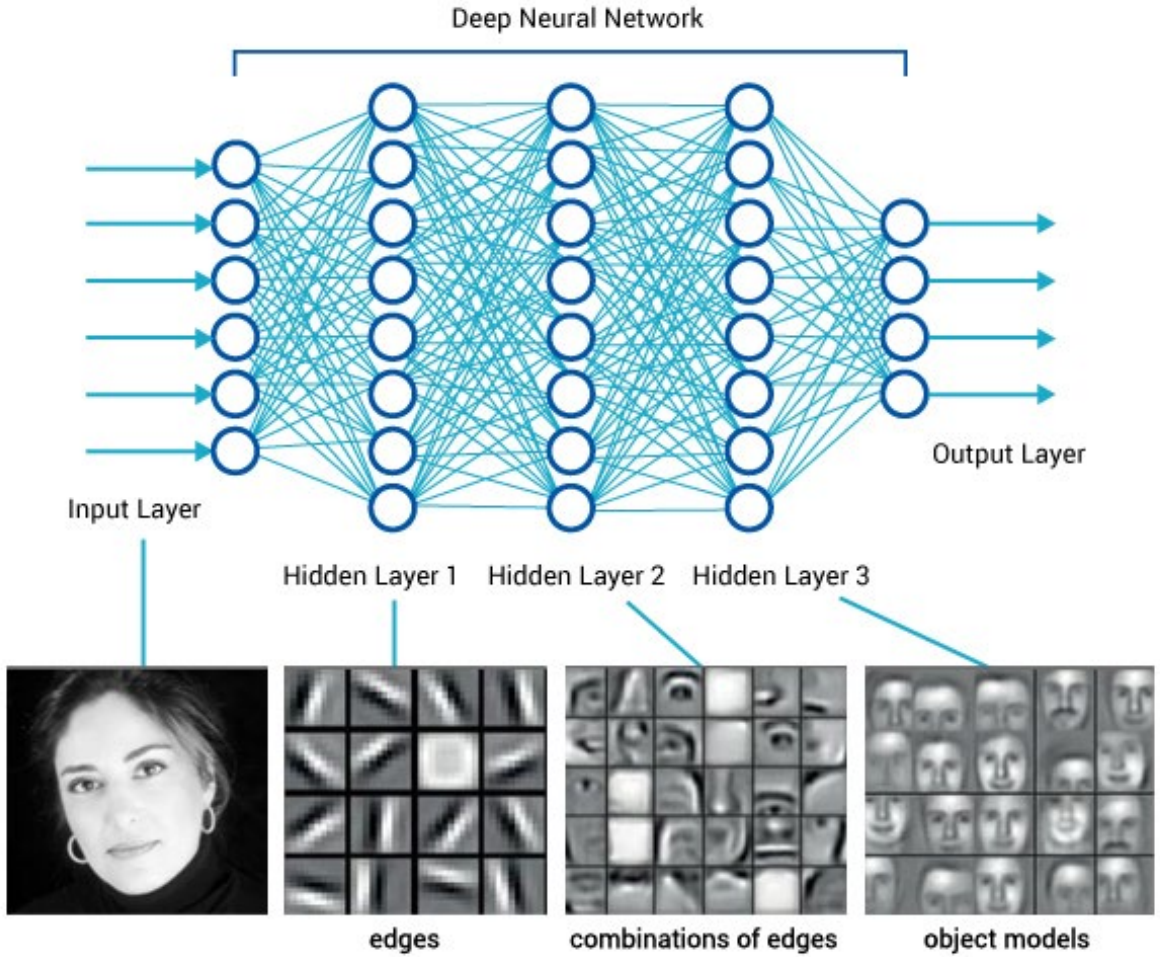
ResNet

Sorry, does not fit in slide.

http://felixlaumon.github.io/assets/kaggle-right-whale/resnet.png

Keras: https://github.com/raghakot/keras-resnet/blob/master/resnet.py

What is happening?



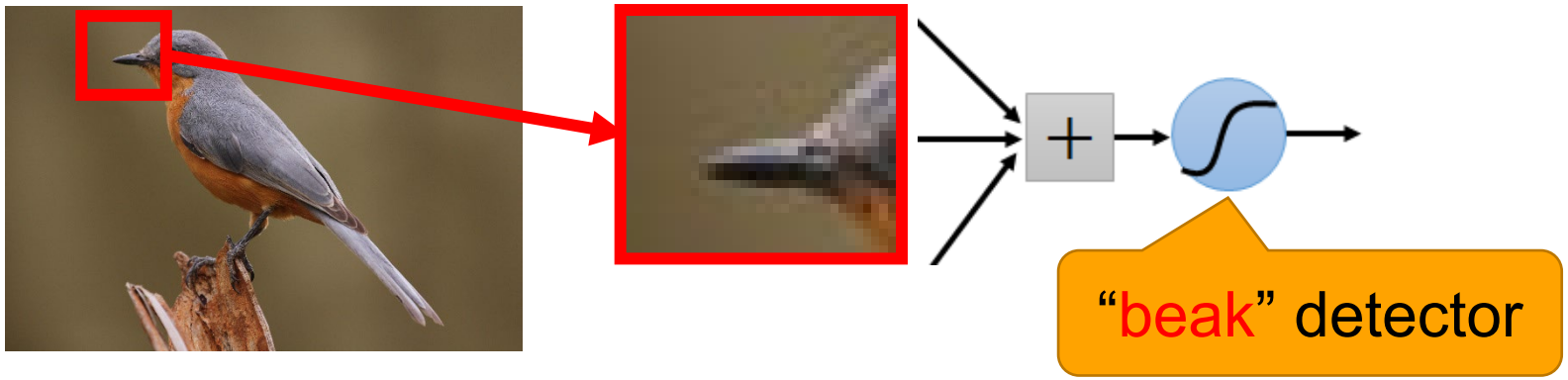
https://www.saagie.com/fr/blog/object-detection-part1

Convolution

Consider learning an image:

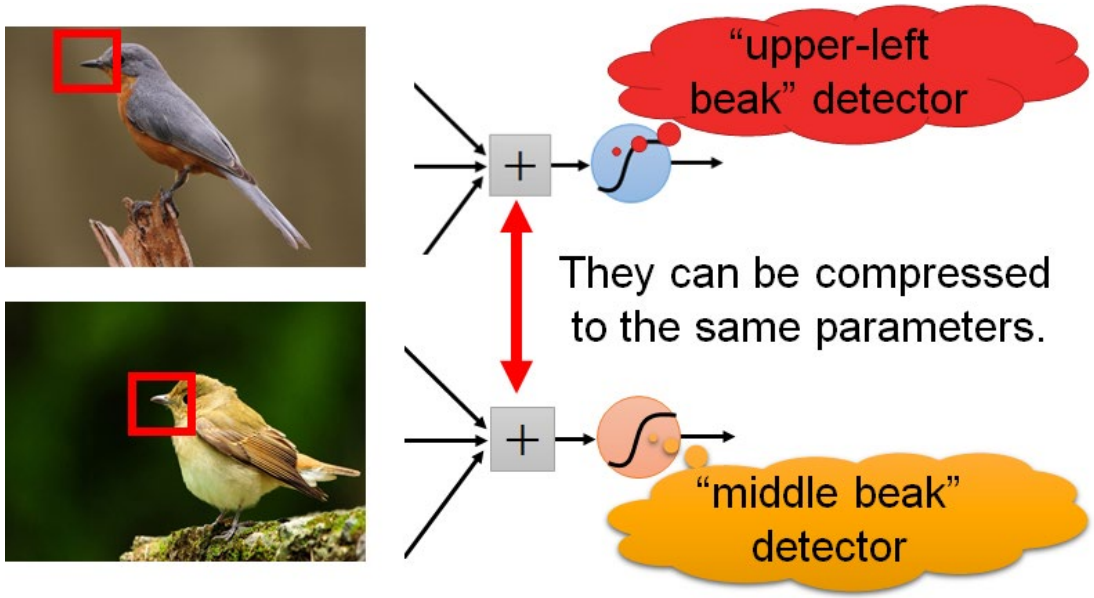
- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters



Detectors

- Same pattern appears in different places:
  They can be compressed!
  What about training a lot of such “small” detectors
  and each detector must “move around”.



Examples CNN

AlphaGo

19 x 19 matrix

Black: 1

white: -1

none: 0

Neural Network

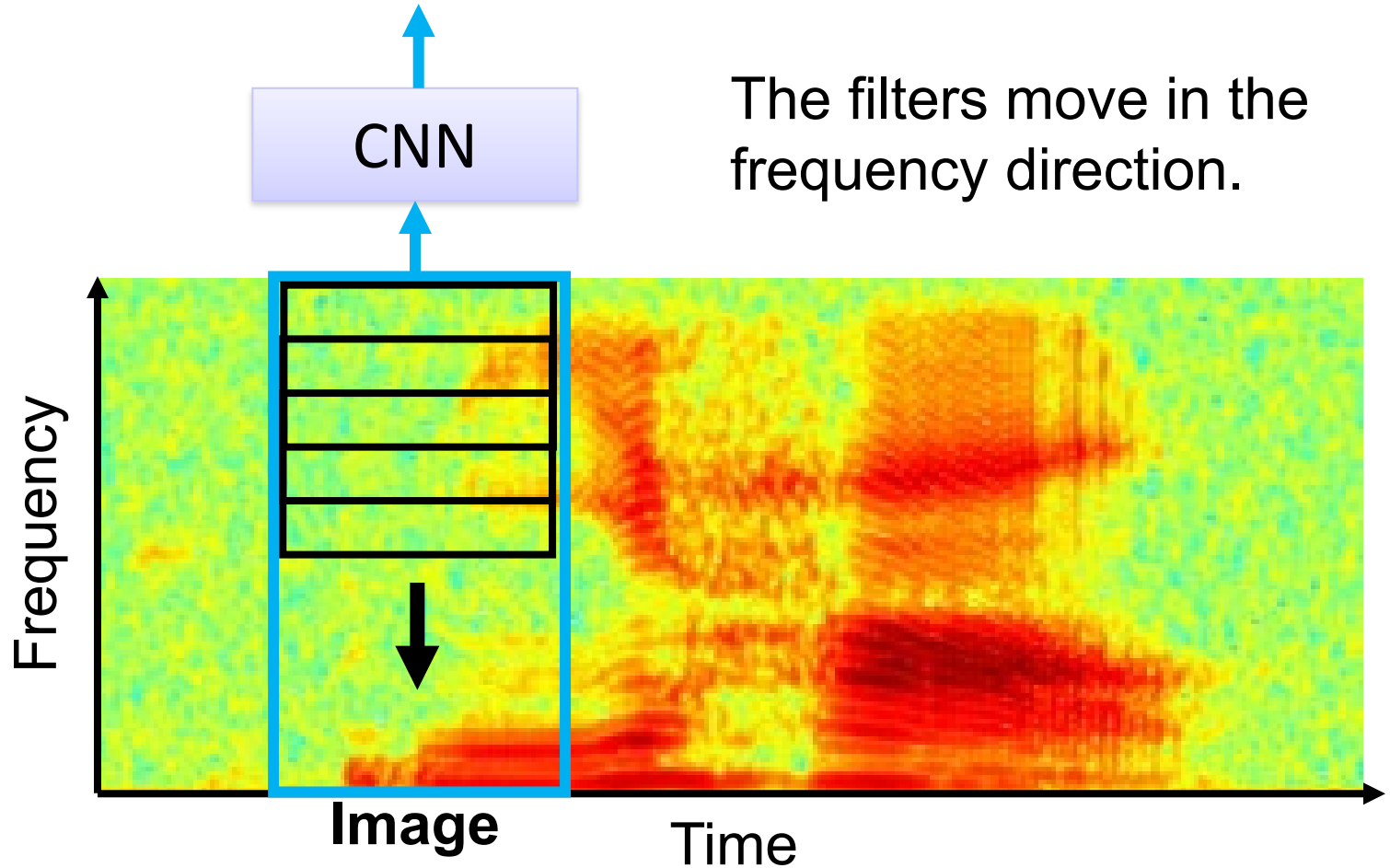
Next move

(19 x 19 positions)

Fully-connected feedforward network can be used

But CNN performs much better

CNN in speech recognition



The filters move in the frequency direction.

Neural Network: Issues

ImageNet

- Not designed for people
- Recently went viral
- Sept 23, 2019
- “ImageNet will remove 600,000 images of people stored on its database after an art project exposed racial bias in the program’s artificial intelligence system.”

ImageNet

- First presented as a research poster in 2009
- Scraped a collection of many millions of images from the internet
- Trained through images categorized by Amazon Mechanical Turk workers
- Crowdsourcing platform through which people can earn money performing small tasks
- Sorted an average of 50 images per minute into thousands of categories
- In 2012, a team from the University of Toronto used a Convolutional Neural Network to handily win the top prize
- Final year 2017, and accuracy in classifying objects in the limited subset had risen from 71.8% to 97.3%. That did not include “Person” category

ImageNet

- AI researcher Kate Crawford and artist Trevor Paglen
  - Training Humans — an exhibition that at the Prada Foundation in Milan
  - Part of their experiment also lives online at ImageNet Roulette, a website where users can upload their own photographs to see how the database might categorize them.
  - https://www.excavating.ai/
- Example of the complexities and dangers of human classification
- The sliding spectrum between supposedly unproblematic labels like “trumpeter” or “tennis player” to concepts like “spastic,” “mulatto,” or “redneck.”
- ImageNet is an object lesson in what happens when people are categorized like objects.

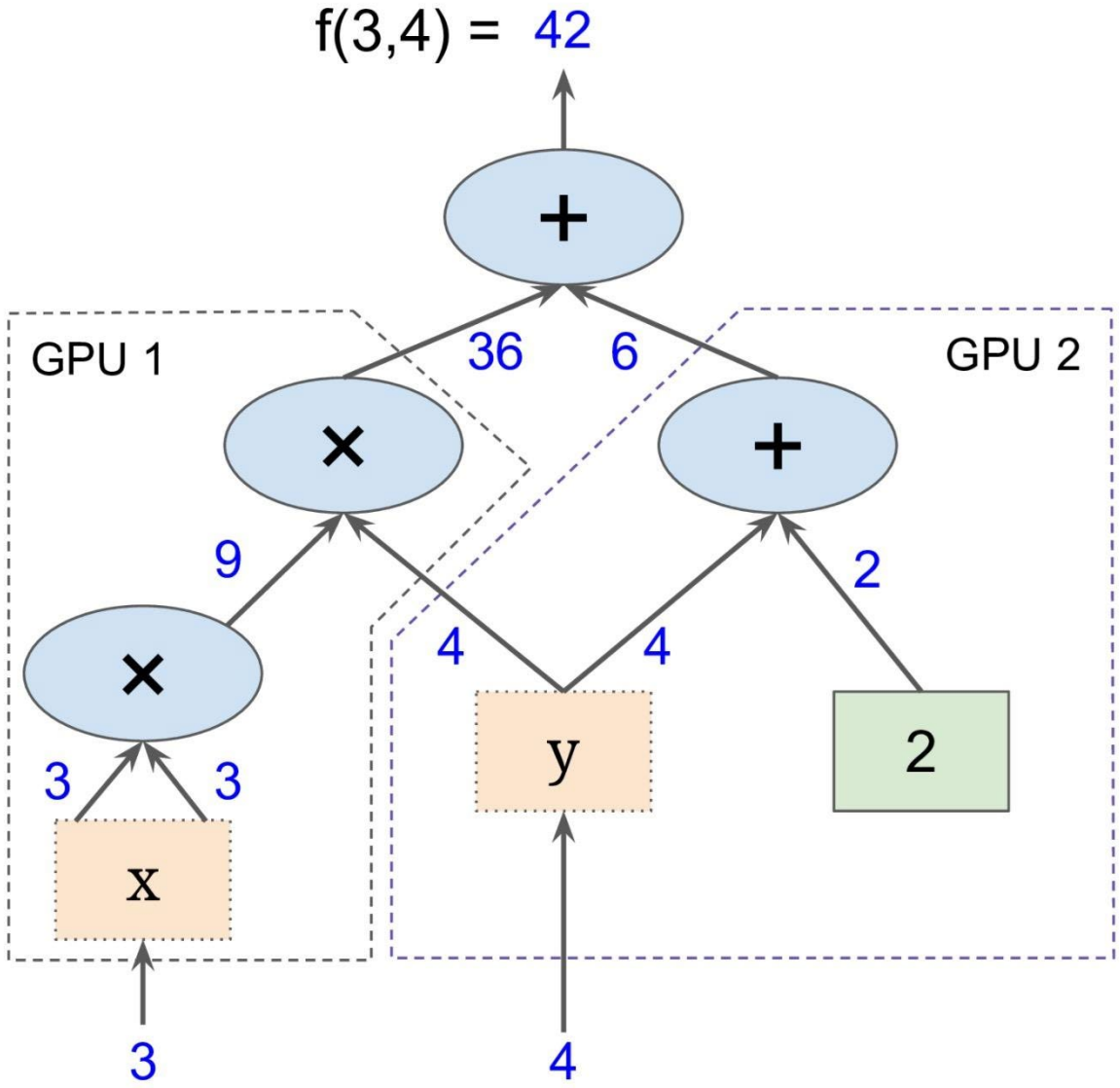
Discussion

Discussion

- Decentralized knowledge representation
  ☞ possibility to parallelize (GPUs!)
- Can find pattern outside of human understanding
- Currently best way to deal with sensory data
- Network structure determines what can be learned
  ☞ must be provided by user
- Represented knowledge not understandable by humans
- Learning can take very long
- Too many learning procedures: when to choose which?

Subgraphs

Possible to break graphs into several chunks ar
multiple CPUs, GPUs, TPUs, or other devices



Graph from *Hands-On Machine Learning with Scikit-Learn and TensorFlow*

Why graphs

1. Save computation. Only run subgraphs that lead to the values you want to fetch.
2. Break computation into small, differential pieces to facilitate auto-differentiation
3. Facilitate distributed computation, spread the work across multiple CPUs, GPUs, TPUs, or other devices
4. Many common machine learning models are taught and visualized as directed graphs

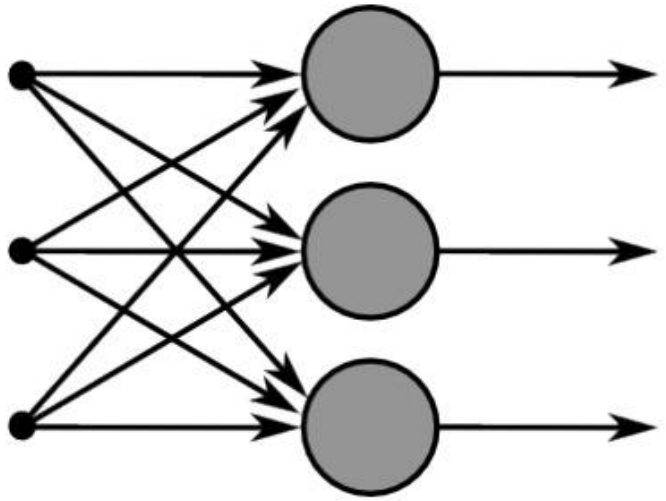


Figure 3: This image captures how multiple sigmoid units are stacked on the right, all of which receive the same input x.

Onward to …
Bias

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~jwhudson/

UNIVERSITY OF CALGARY