Carnegie Mellon University

Introduction to Deep Learning for Engineers

Spring 25, Introduction to Deep Learning
Jan 14, 2025

Amir Barati Farimani

Russel V. Trader Associate Professor of Mechanical Engineering and Bio-Engineering

Carnegie Mellon University

Text To Image Generation

DALL. E2

“a teddy bear on a skateboard in times square”

“Hierarchical Text-Conditional Image Generation with CLIP Latents” Ramesh et al.,2022

Imagen

- A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk.

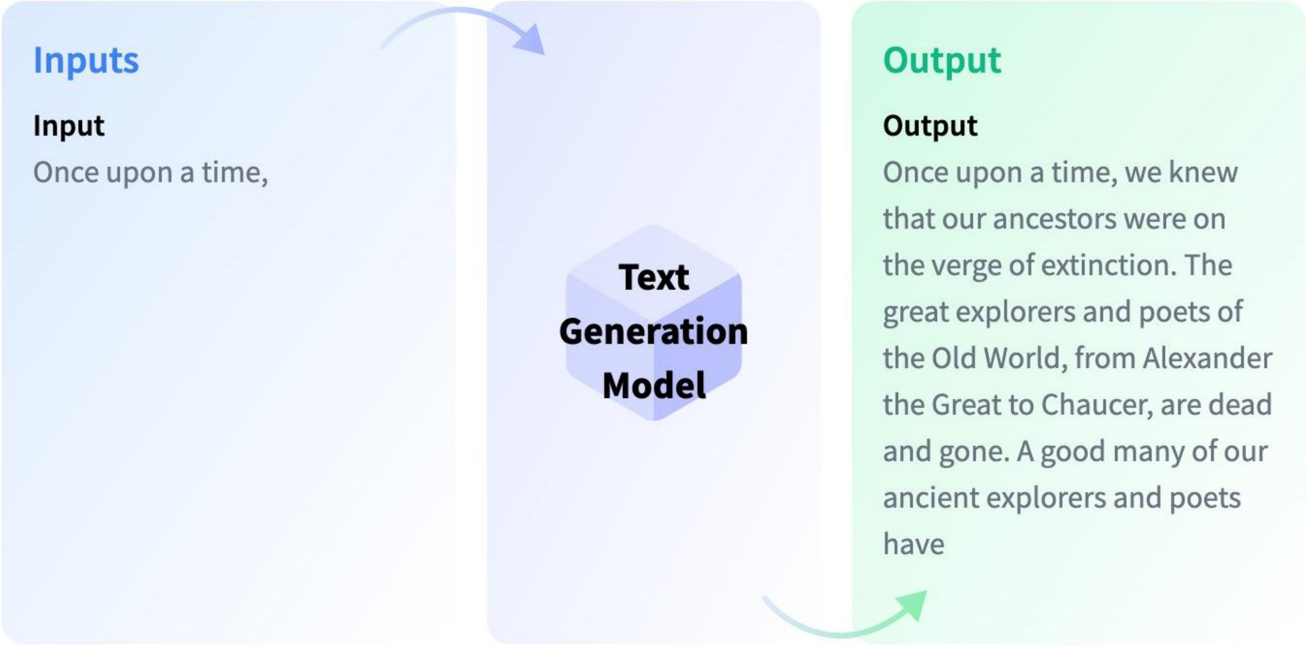
“Photorealistic Tex-to-Image Diffusion Models with Deep Language Understanding”, Saharia et al.,2022

Text Generation

Natural Language Generation (NLG)

- Language modeling
- Conditional language modeling

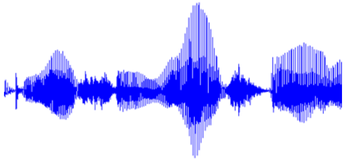
Next word prediction



https://huggingface.co/tasks/text-generation

What do we want computers to do with our data?

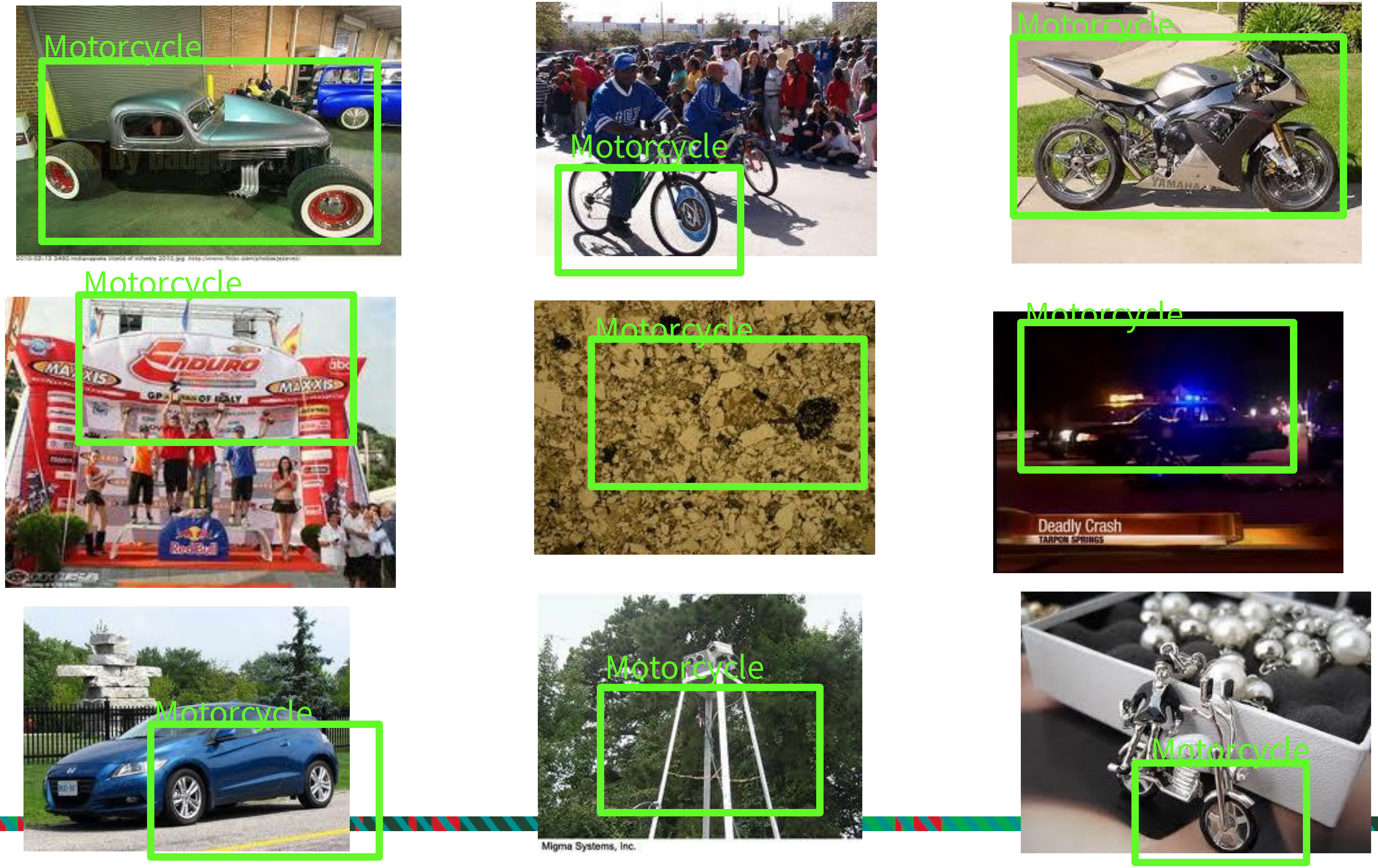
Label: “Motorcycle”
Suggest tags
Image search
…



Speech recognition
Music classification
Speaker identification
…

Web search
Anti-spam
Machine translation
…

Computer vision is hard!



Why is this hard?

You see this:

But the camera sees this:

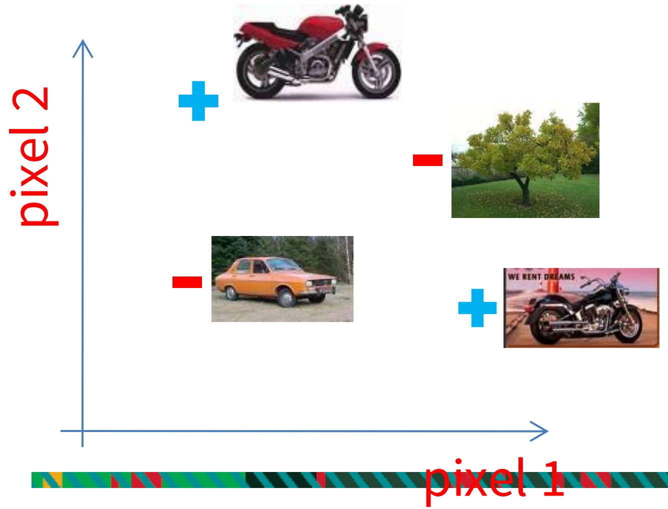
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Machine learning and feature representations

Input

Raw image

Learning algorithm



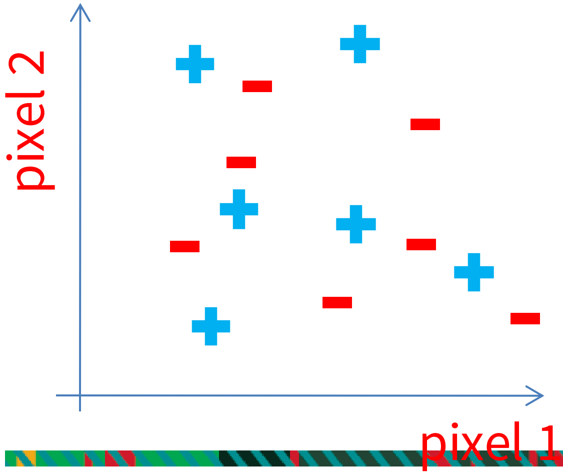
+ Motorbikes
– "Non"-Motorbikes

What we want

Input → **Feature representation** → **Learning algorithm**

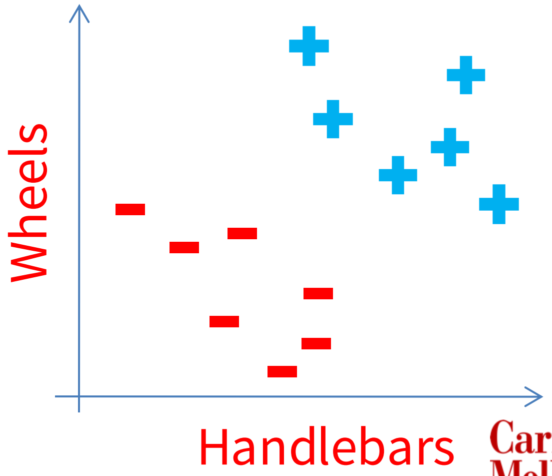
E.g., Does it have Handlebars? Wheels?

Raw image



- ✚ Motorbikes
- – “Non”-Motorbikes

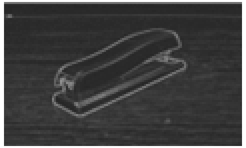
Features



How is computer perception done?

Images/video

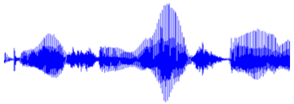
Image



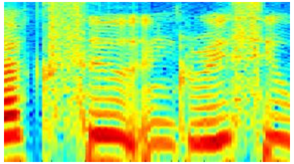
Vision features

Detection

Audio



Audio

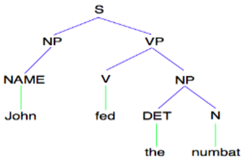


Audio features

Speaker ID

Text

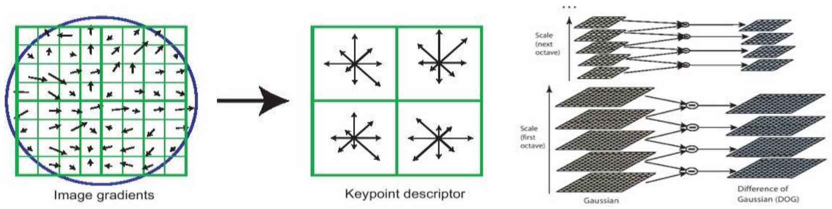
Text



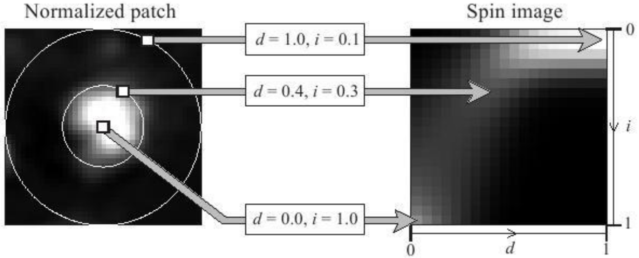
Text features

Text classification,
Machine translation,
Information retrieval,
....

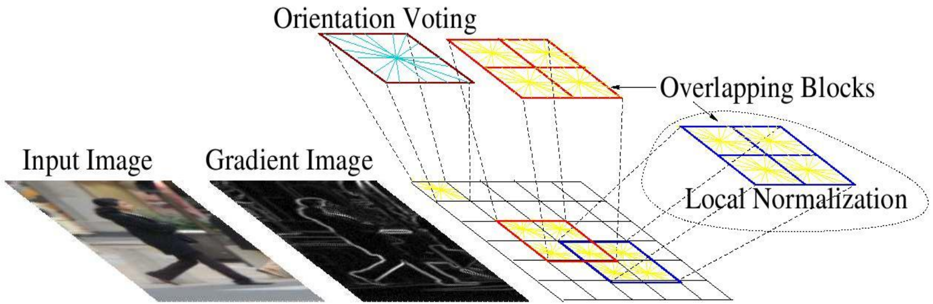
Computer vision features



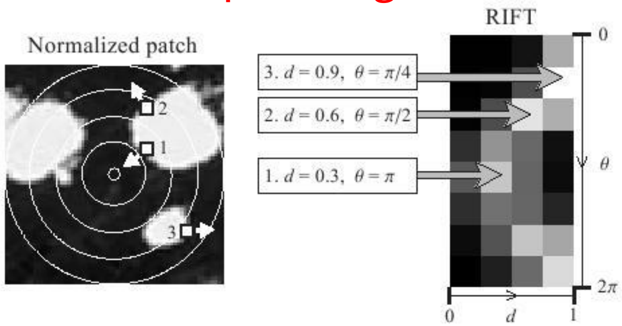
SIFT



Spin image



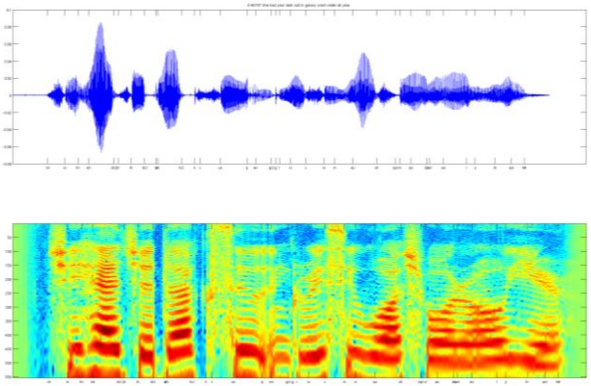
HoG



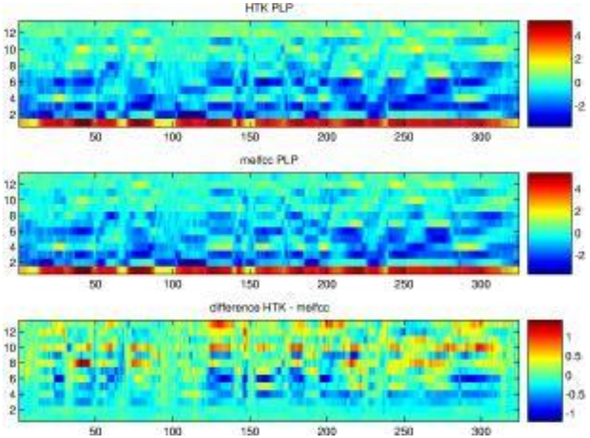
RIFT

Carnegie Mellon University

Audio features

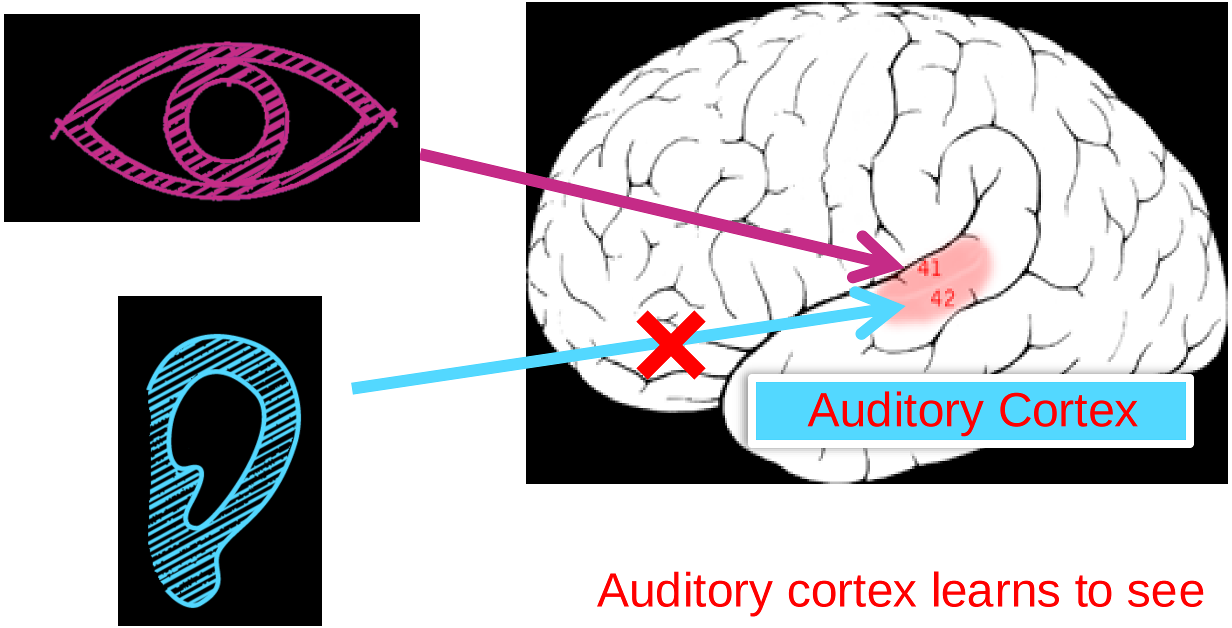


Spectrogram



MFCC

The “one learning algorithm” hypothesis



Auditory cortex learns to see

[Roe et al., 1992]

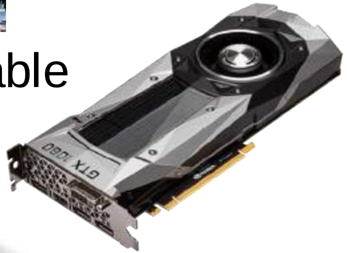
We Need a Machine Learning Universal Representation Learning Method

The Answer: Deep Learning

Why Now?

- **Big Data:** Appearance of big datasets and challenges such as ImageNET

- **GPU and Hardware Accelerator:** Massively scalable and Parallelizable



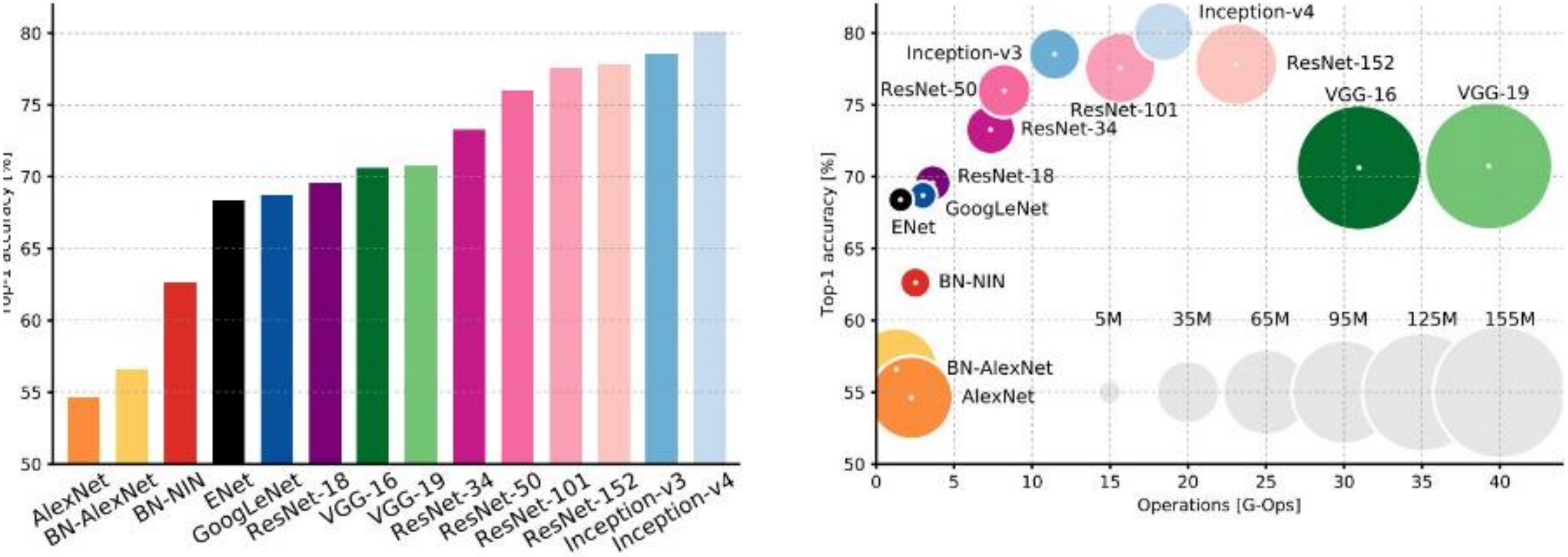
- **Algorithms:** Optimization, Drop out, Batch Normalization,…

Software: Optimization, Drop out, Batch Normalization,…

TensorFlow

Carnegie Mellon University

Why Now?

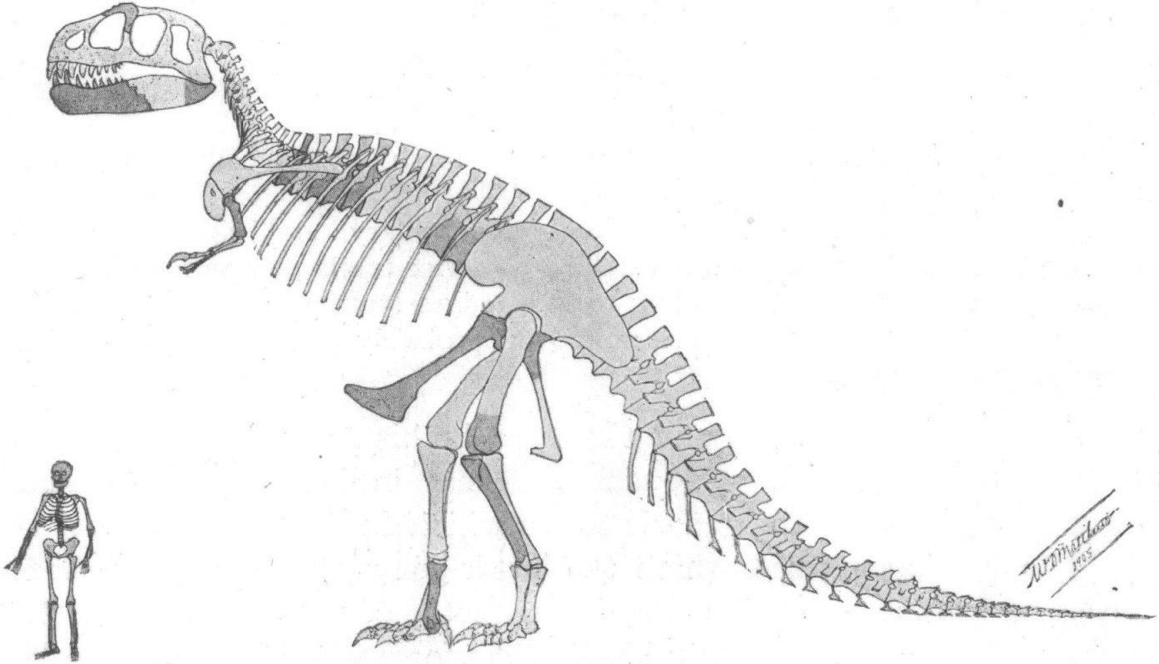


An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Carnegie Mellon University

Foundational Models

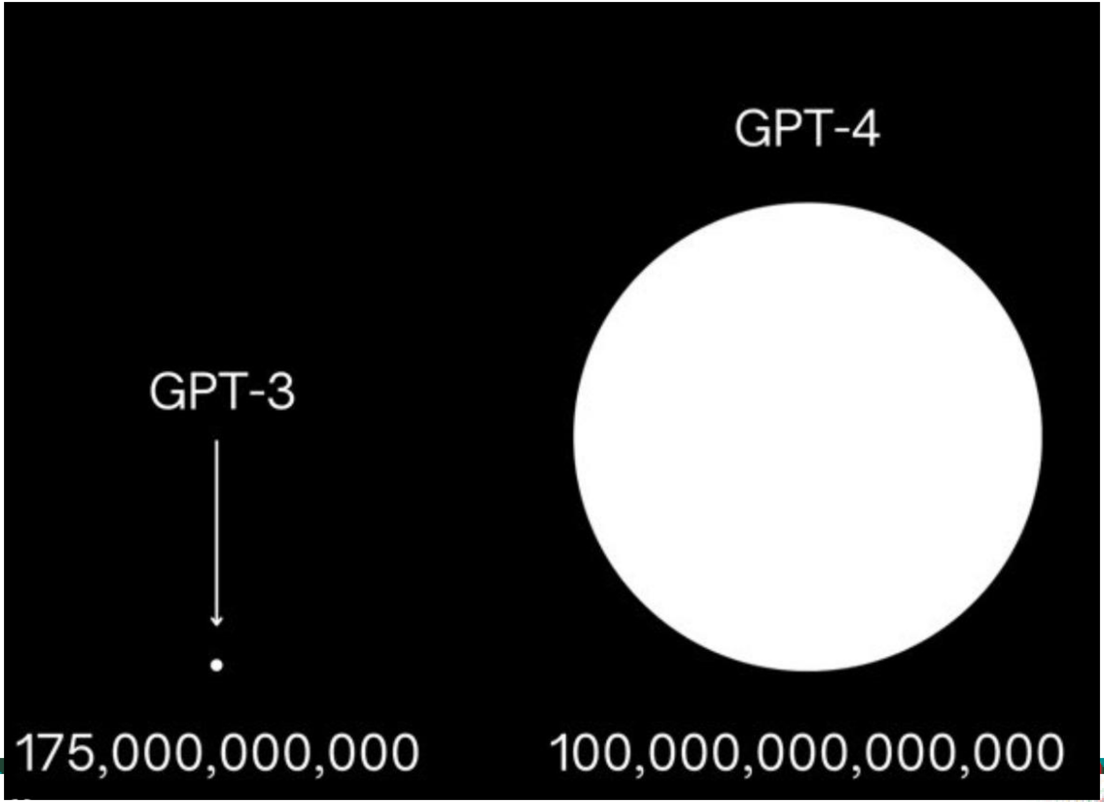
Foundational Models



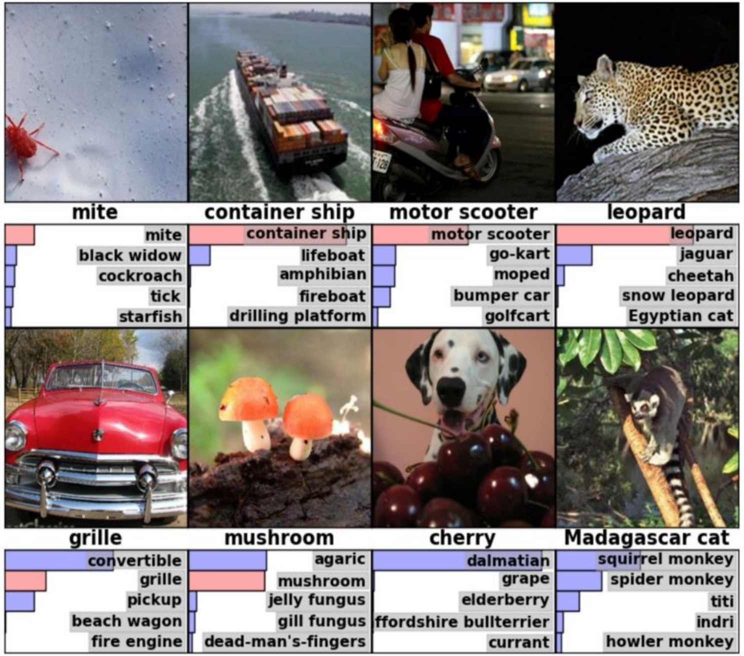
GPT-2
1.5B Parameters

GPT-3
175B Parameters

Foundational Models

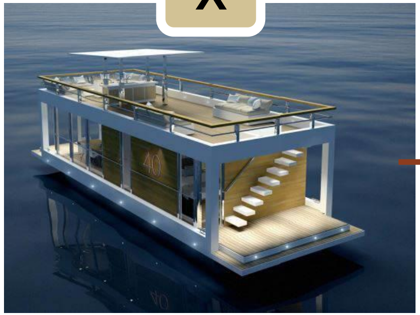


Exuberance of AI



X

Y

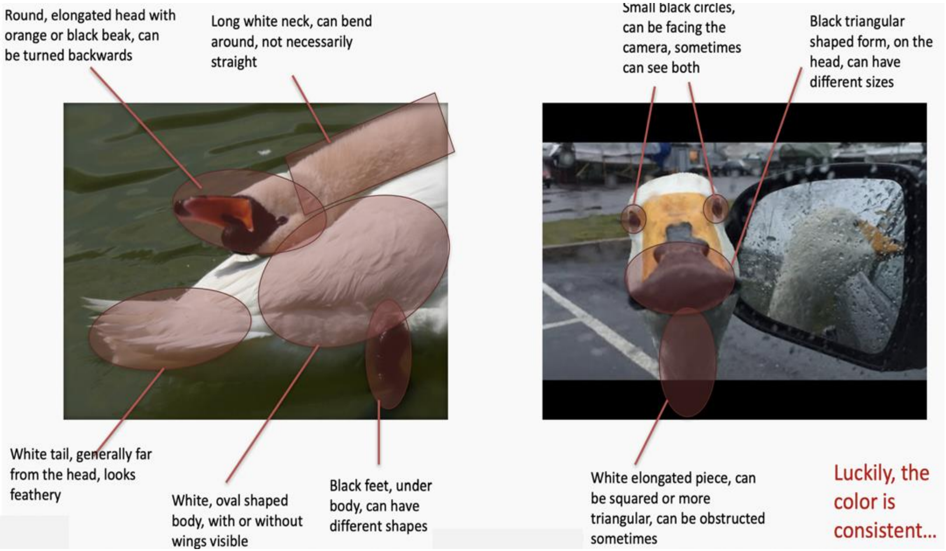


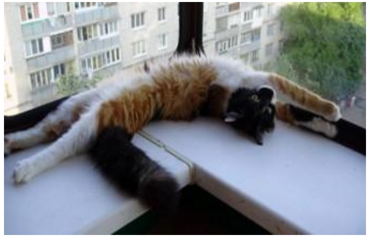
Houseboat

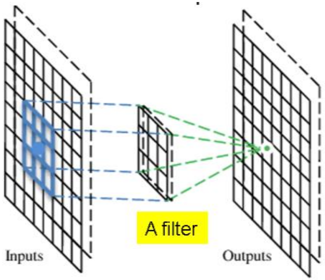
Boat

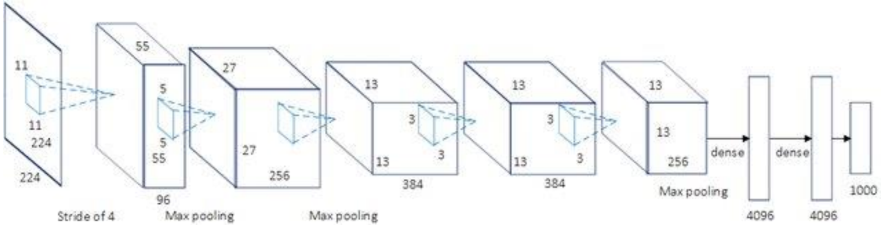
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

Revolution 1: Automatic Feature Detection (2012)





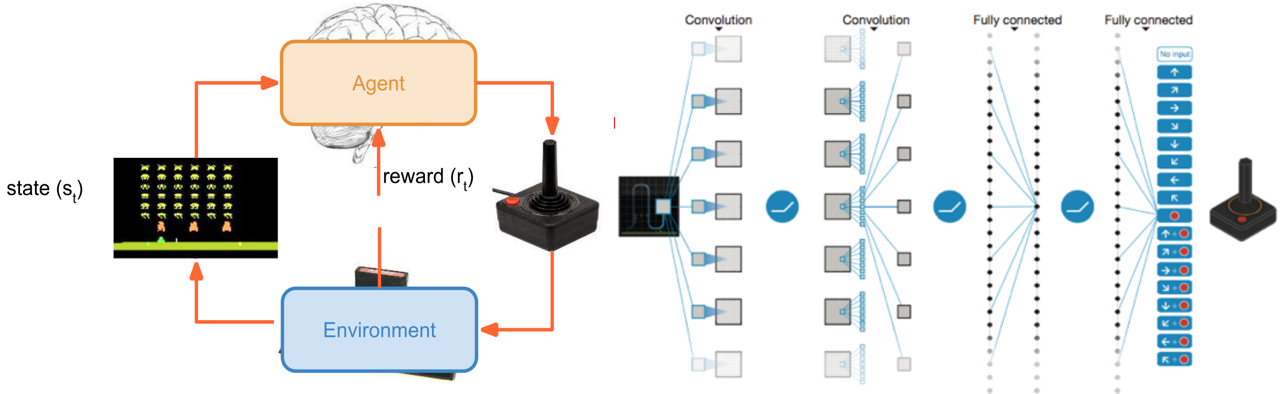


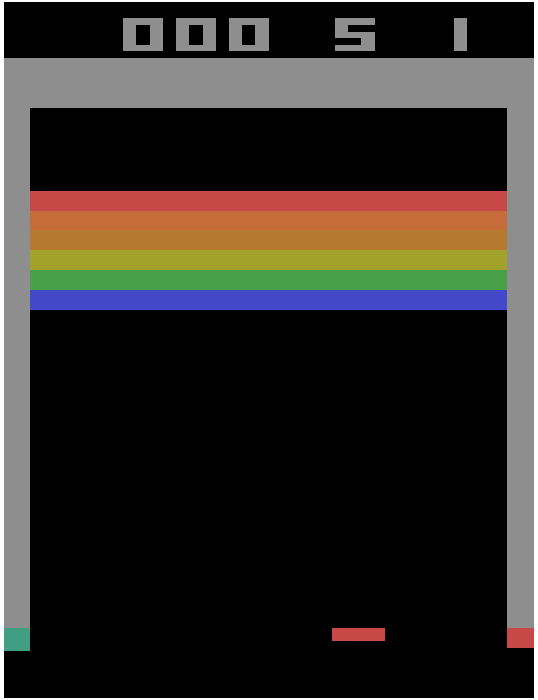


Carnegie Mellon University

Revolution 2: Deep Reinforcement Learning (2015)

Q-learning + Function Approximation + Deep Network





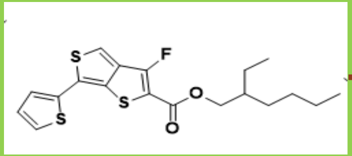
Human-level control through deep reinforcement learning, Nature, 518, pages, 529–533, (2015)

Carnegie Mellon University

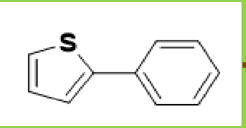
Why AI and ML?

X

Y (Solubility)

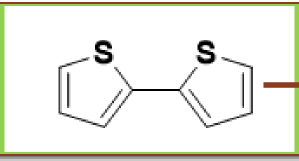


0.71



0.53

Training



?

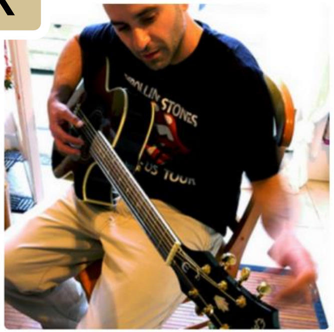
Test

X

"construction worker in orange safety vest is working on road."

Y

X

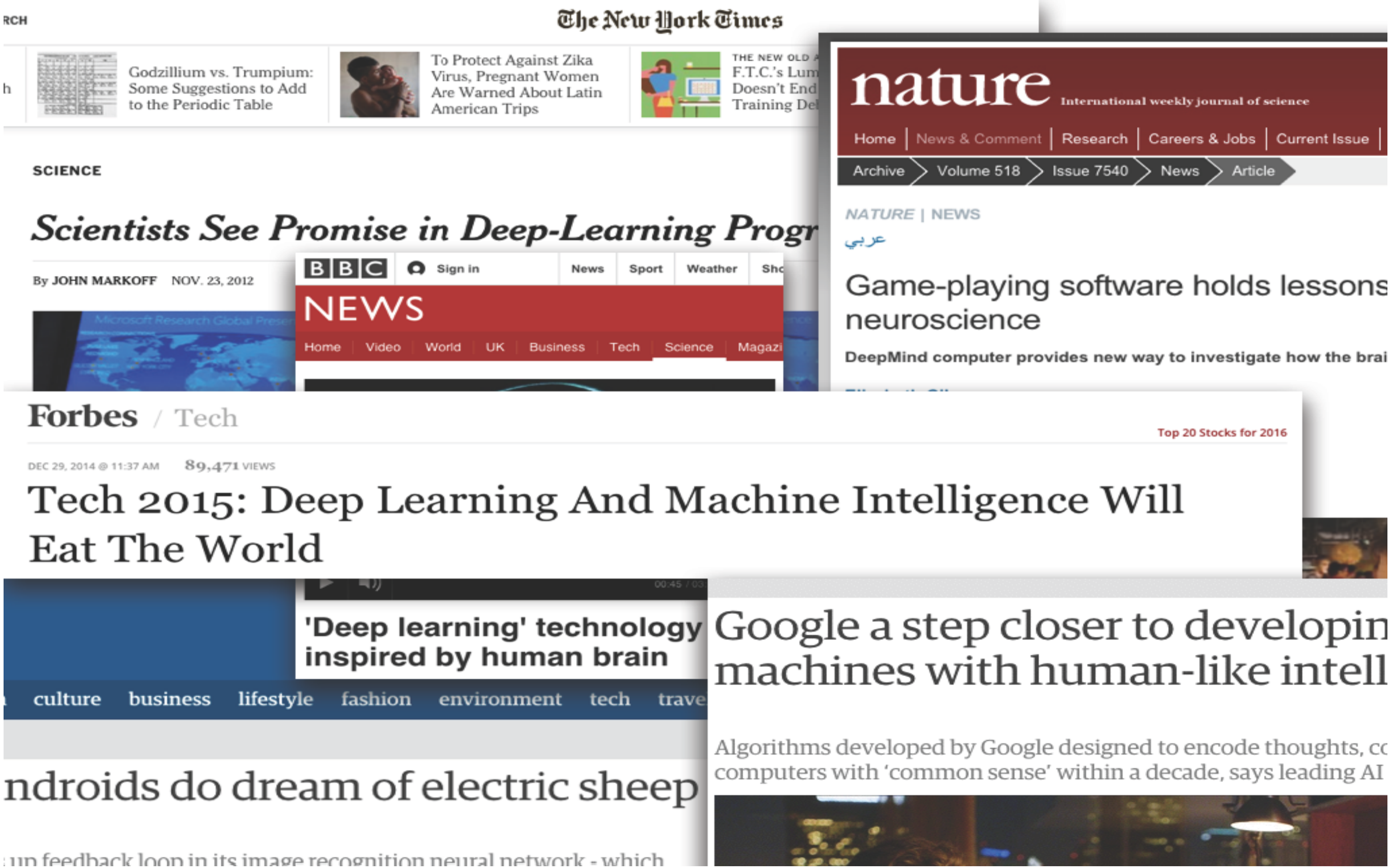


"man in black shirt is playing guitar."

Y

X

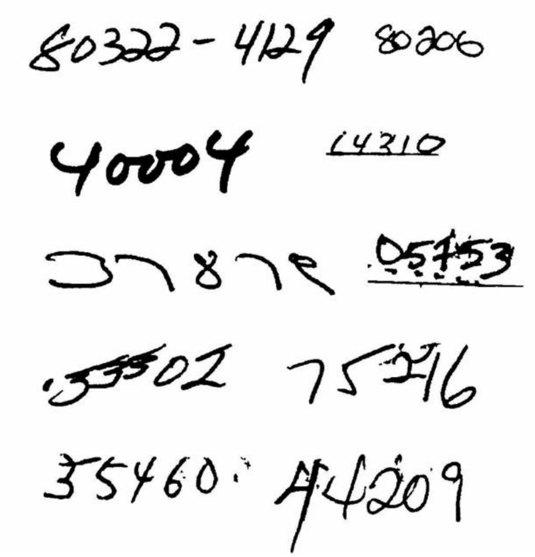
?

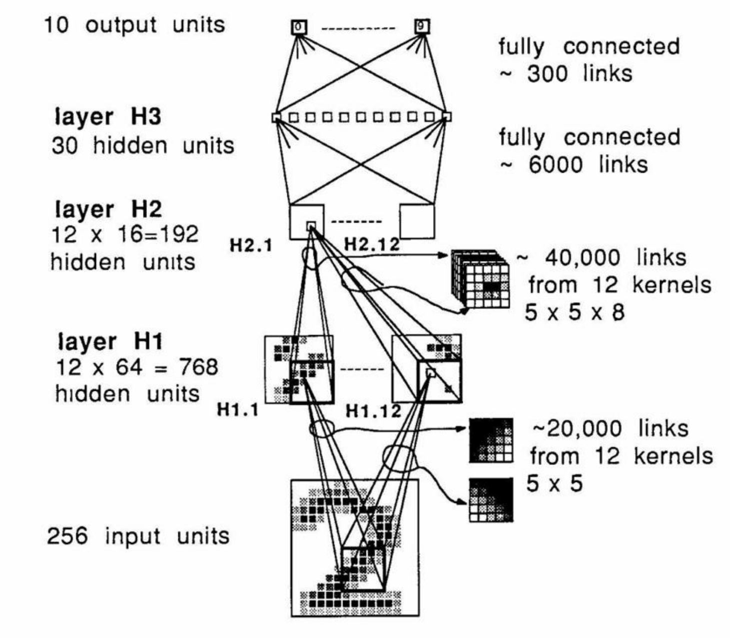


Carnegie Mellon University

Milestone 1: Digit Recognition

LeNet 1989: recognize zip codes, Yann Lecun, Bernhard Boser and others, ran live in US postal service

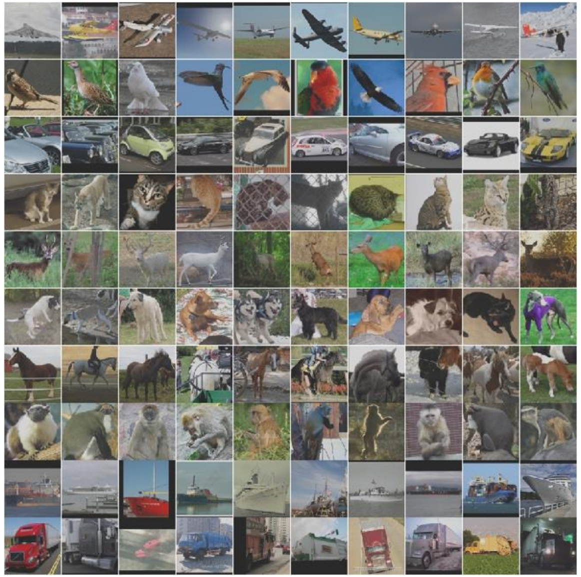


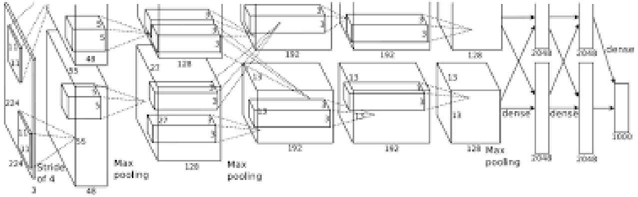


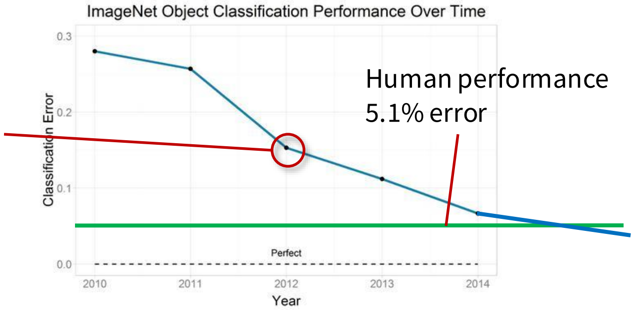
Carnegie Mellon University

Milestone 2: Image Classification

Convolutional NNs: AlexNet (2012): trained on 200 GB of ImageNet Data

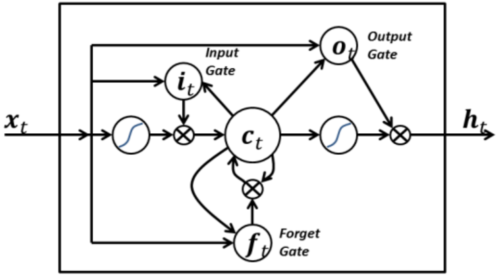


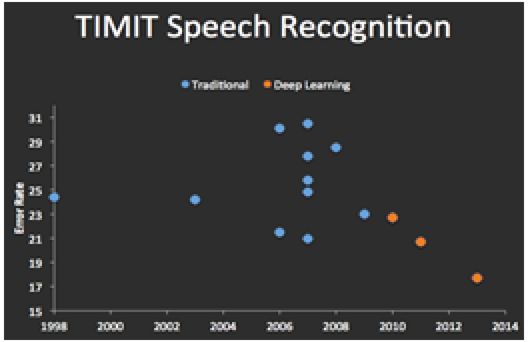


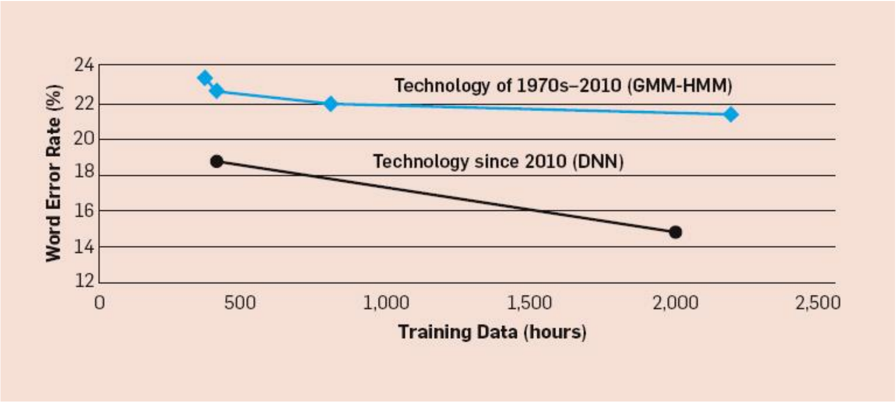


Milestone 3: Speech Recognition

Recurrent Nets: LSTMs (1997):

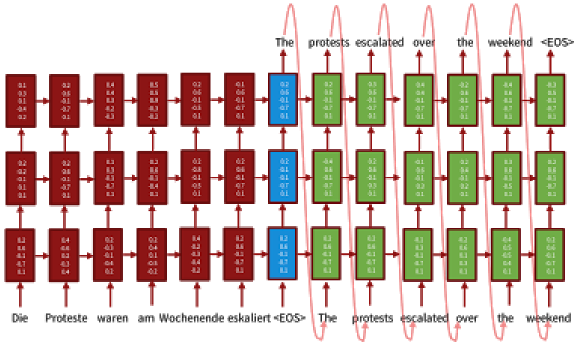


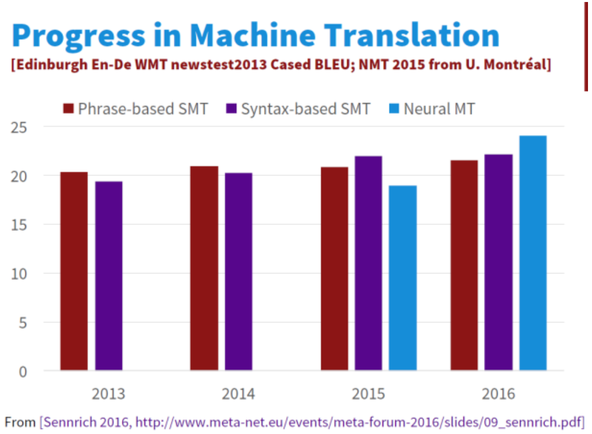




Milestone 4: Machine Translation

Sequence-to-sequence models with LSTMs and attention:

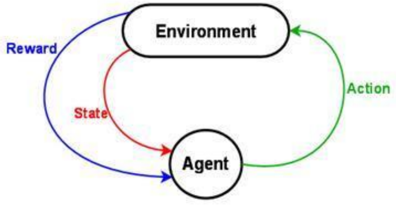


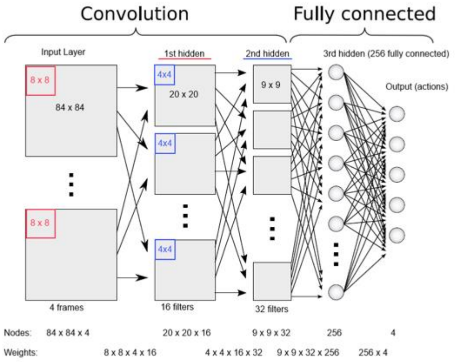


Source Luong, Cho, Manning ACL Tutorial 2016.

Milestone 5: Deep Reinforcement Learning

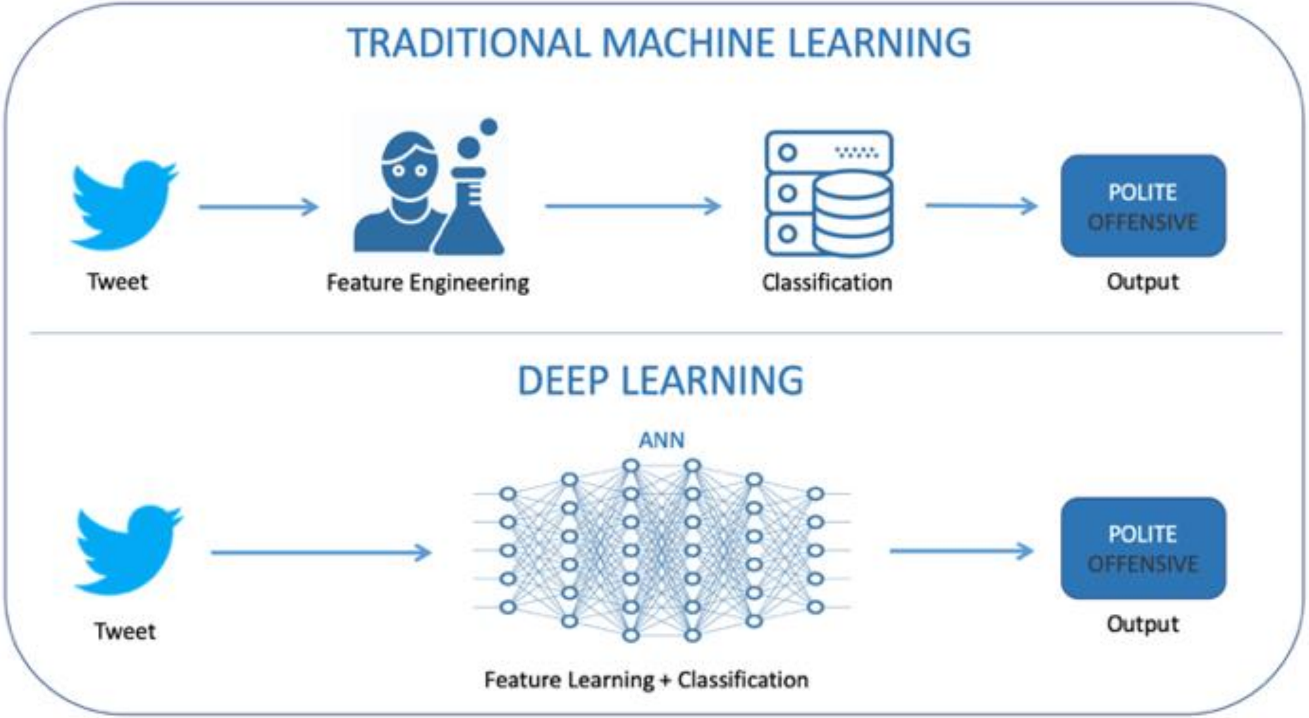
In 2013, Deep Mind's arcade player bests human expert on six Atari Games. Acquired by Google in 2014,.





In 2016, Deep Mind's AlphaGo defeats former world champion Lee Sedol

Traditional ML vs Deep Learning



Carnegie Mellon University

How Engineers Can use AI?

- Complex Phenomena in Engineering that we don't have Mathematical Description
- Where we don't know the Physics, but we want to predict to engineer
- Where we want to speed up computation by surrogate, fast models
- Where we have abundance of data from experiments and want to make
a more precise model

Fluid Mechanics

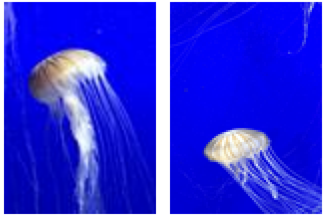
Solid Mechanics

Bio-Engineering

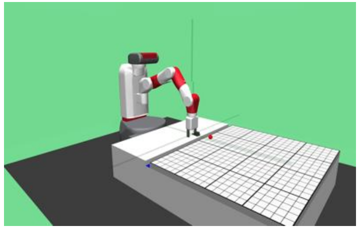
Dynamic and Control

Material Science

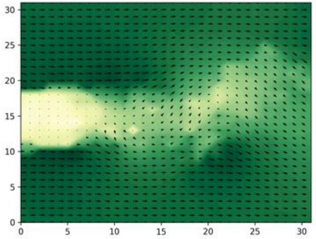
System Identification



Robotics

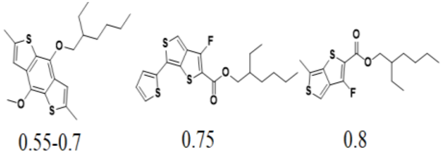


Fluid Mechanic: Turbulence



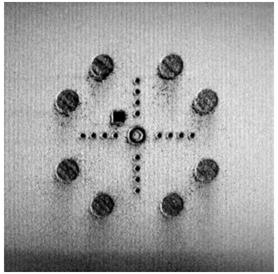
Manufacturing: Process Control

Material Property Prediction

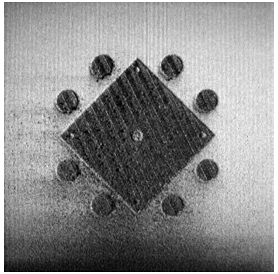


Carnegie Mellon University

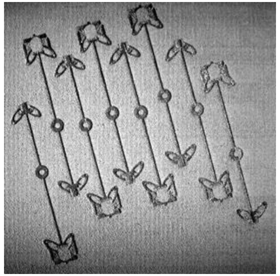
Immediate Use of AI in Engineering
Where images/vision exists



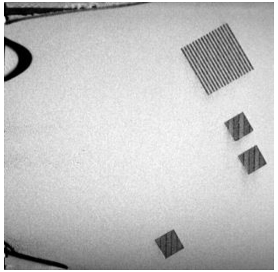
Actual - 2
Predicted - 1



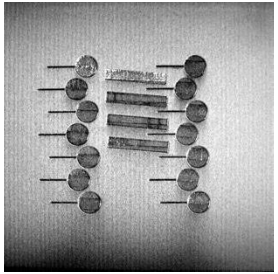
Actual - 2
Predicted - 1



Actual - 0
Predicted - 1



Actual - 1
Predicted - 1

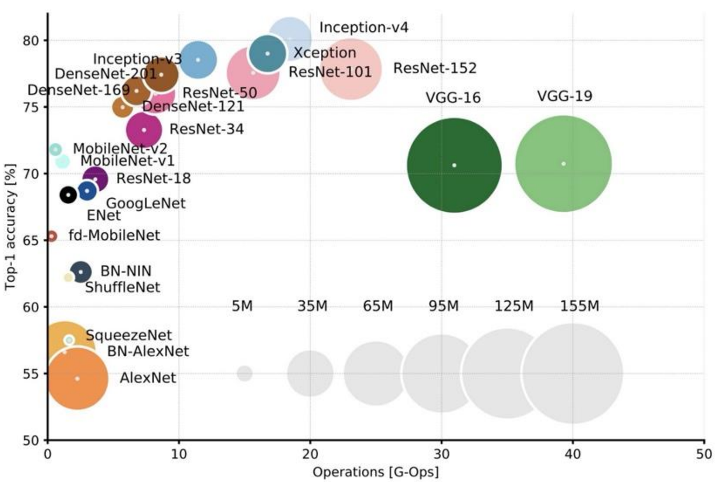


Actual - 2
Predicted - 2

Actual - 1
Predicted - 1

CNN Architecture	% Accuracy
VGG16	80.6
ResNet18	80.9
ResNext50	81.3
Dense Net	81.7
Custom CNN	82.1

‘0’ – No spatter
‘1’ – Some spatter
‘2’ – Too much of spatter



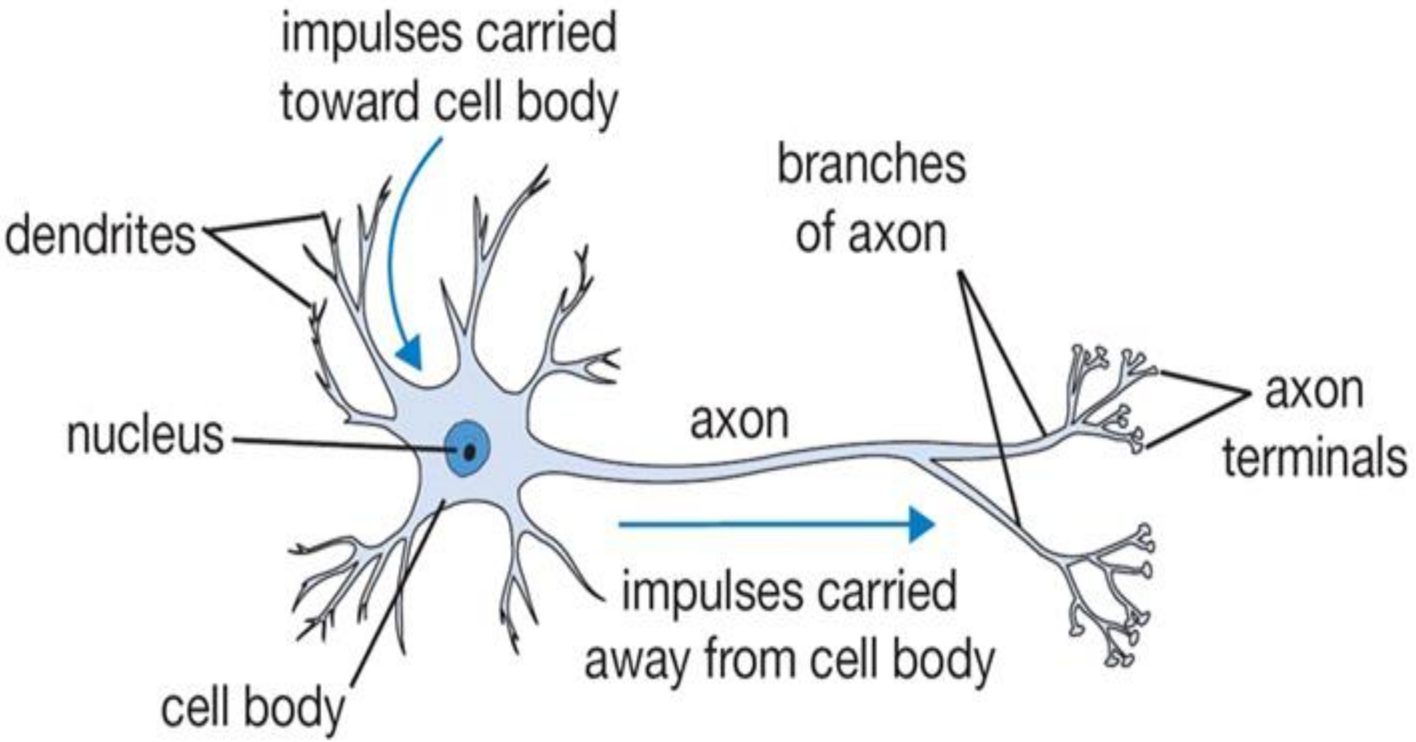
https://towardsdatascience.com

Course Objectives

This course provides an introduction to deep learning. We will learn about the basics of deep neural networks, and their applications to different tasks in engineering. Students will be able to **apply deep learning** to a variety of artificial intelligence tasks pertinent to different engineering problems. Applications of deep learning in Mechanical, chemical, biological, Electrical and material engineering will be discussed.

Biological Neural Network

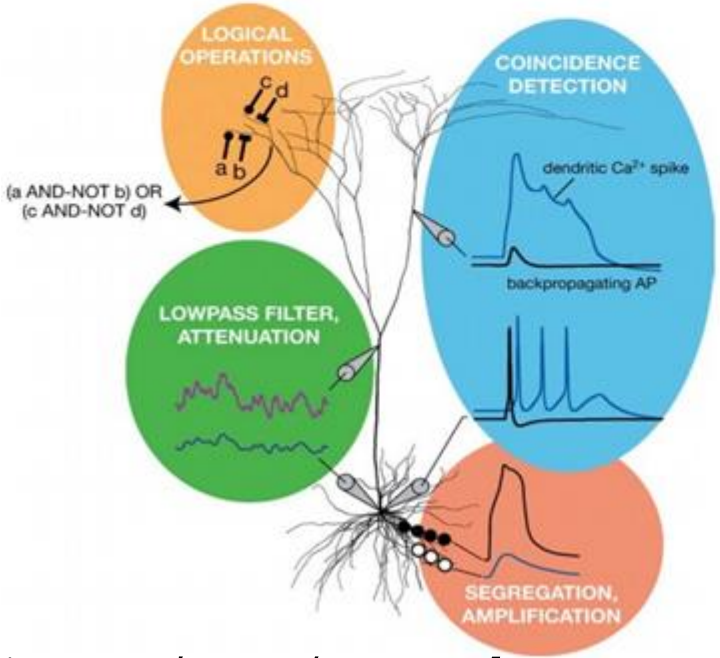
Natural Neuron



Operation of a Biological Neural Network

Biological Neurons:

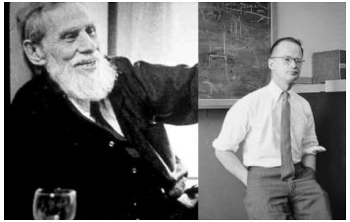
- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate



[Dendritic Computation. London and Hausser]

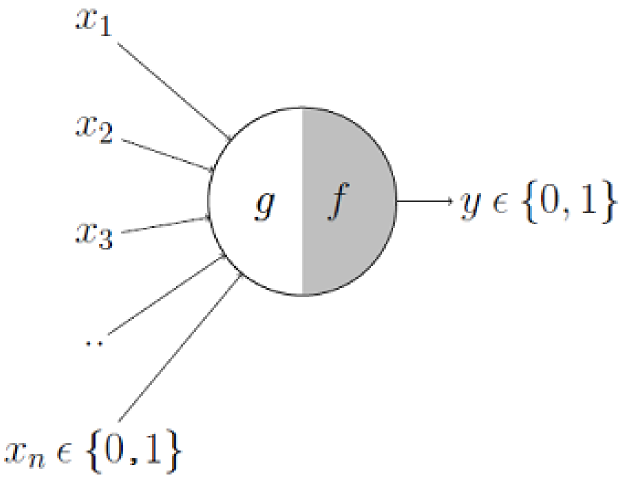
Carnegie Mellon University

McCulloch-Pitts Neuron



- The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.

- Boolean Functions Using M-P Neuron
- The inputs are all Boolean i.e., {0,1}
- The outputs are all Boolean



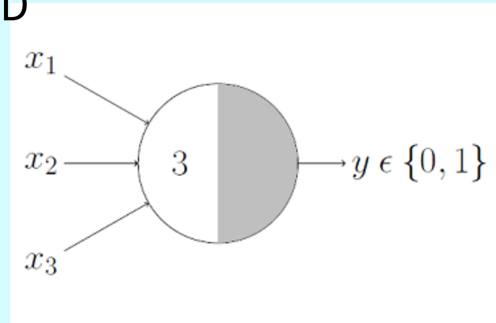
$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

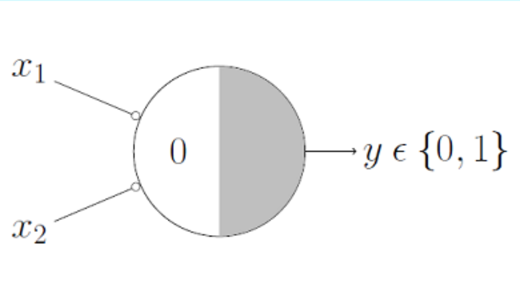
Carnegie Mellon University

McCulloch-Pitts as Boolean function

AND

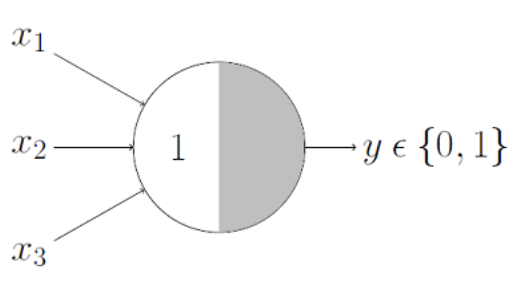


$\boldsymbol{g}(\mathbf{x}) \geq 3$



$\boldsymbol{g}(\mathbf{x}) = 0$

OR



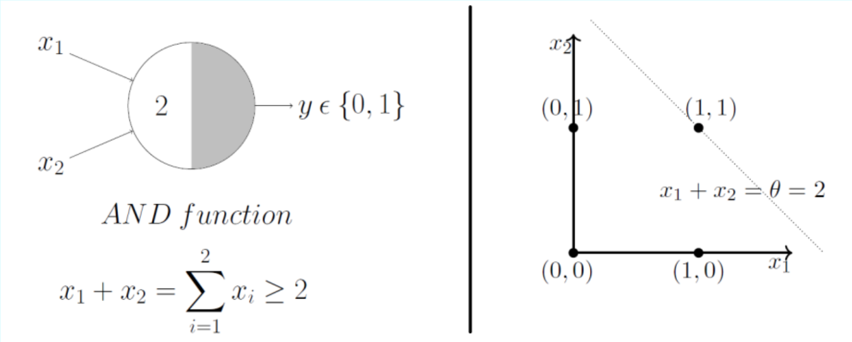
$\boldsymbol{g}(\mathbf{x}) \geq 1$

Note: all the inputs and outputs are {0,1}

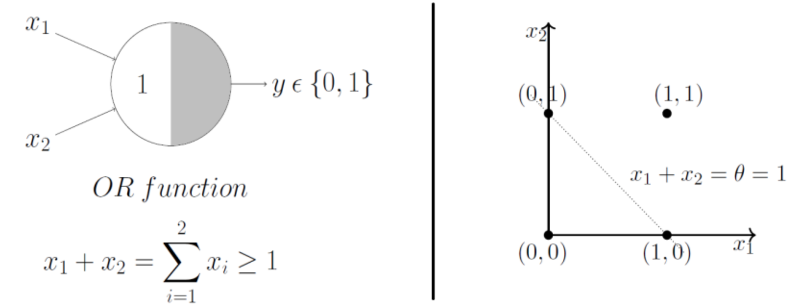
Carnegie Mellon University

Geometrical Understanding of MP

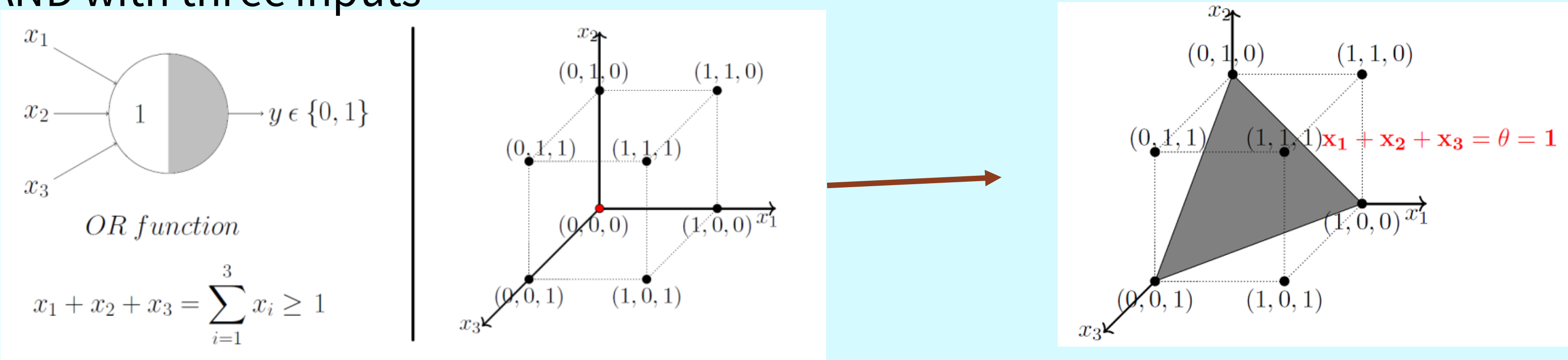
AND with two inputs



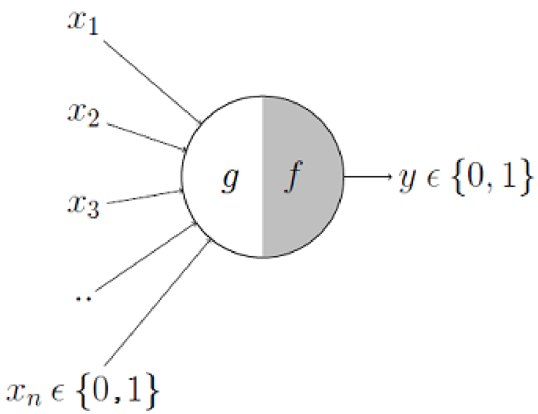
OR with two inputs



AND with three inputs



Limitations of M-P Neuron



What about non-boolean (say, real) inputs?

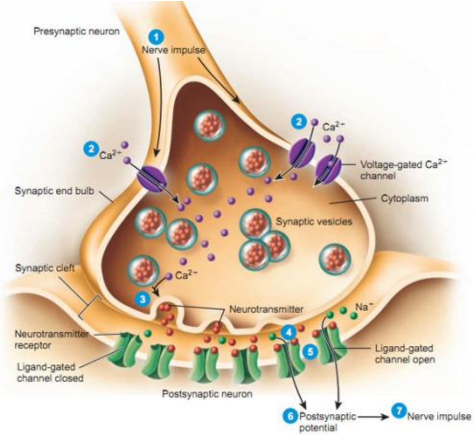
Do we always need to hand code the threshold?

Are all inputs equal? What if we want to assign more importance to some inputs?

What about functions which are not linearly separable? Say XOR function.

Hebbian Learning

The Hebbian rule was the first learning rule. In 1949 Donald Hebb developed it as learning algorithm of the unsupervised neural network. We can use it to identify how to improve the weights of nodes of a network.



- The Hebb learning rule assumes that:

- If two neighbor neurons activated and deactivated at the same time.

Then the weight connecting these neurons should increase.

- For neurons operating in the opposite phase, the weight between them should decrease.
- If there is no signal correlation, the weight should not change.

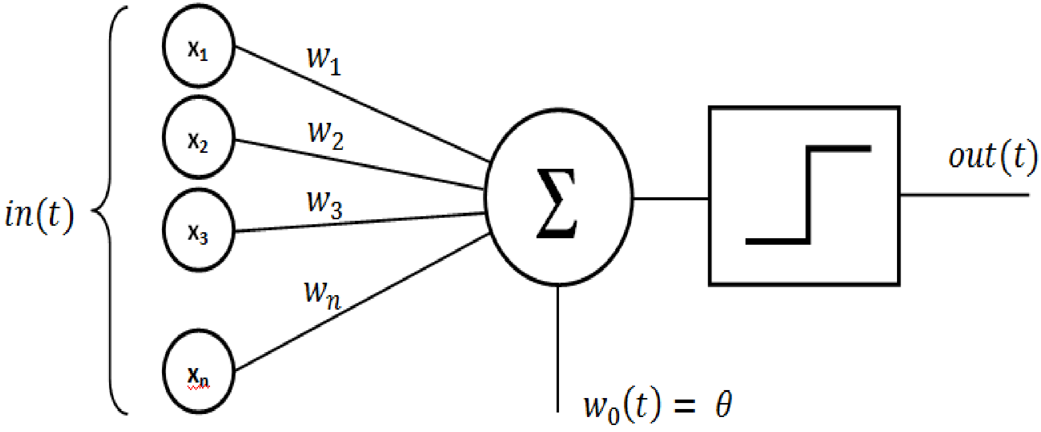
Carnegie Mellon University

Perceptron

Frank Rosenblatt

– Psychologist,Logician

– Perceptron Inventor (1958)



Perceptron Algorithm

Algorithm 1: Perceptron Learning Algorithm

Input: Training examples $\{\mathbf{x}_i, y_i\}_{i=1}^{m}$.

Initialize $\mathbf{w}$ and b randomly.

while *not converged* **do**

    # # # Loop through the examples.
    for $j = 1, m$ **do**

        # # # Compare the true label and the prediction.
        $error = y_j - \sigma(\mathbf{w}^T\mathbf{x}_j + b)$

        ### If the model wrongly predicts the class, we update the weights and bias.
        if *error != 0* **then**

            ### Update the weights.
            $\mathbf{w} = \mathbf{w} + error \times x_j$

            ### Update the bias.
            $b = b + error$

    Test for convergence

Output: Set of weights $\mathbf{w}$ and bias b for the perceptron.

Carnegie Mellon University

Perceptron

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

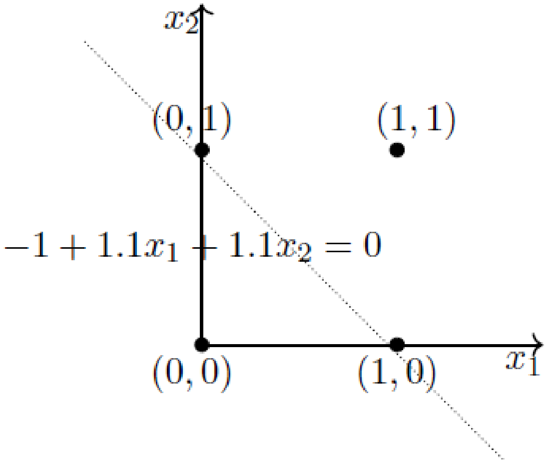
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 > -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 > -w_0$$

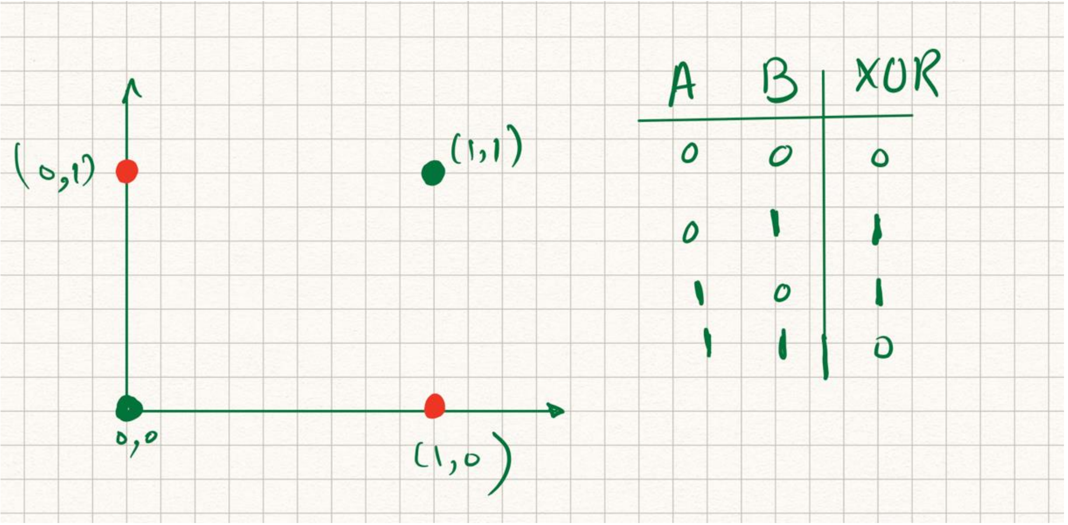
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 > -w_0$$

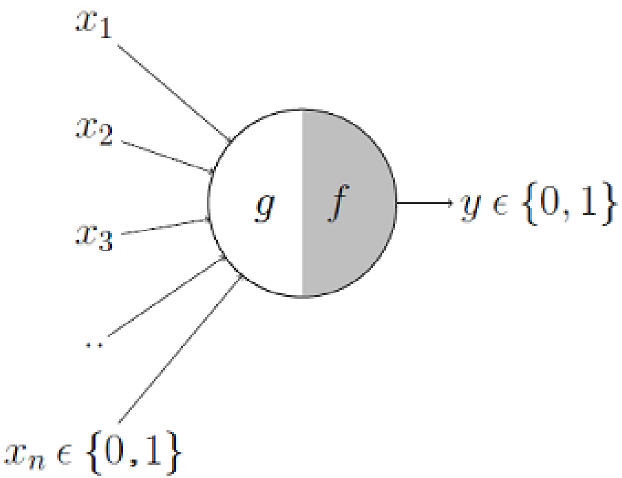
One possible solution is

$$w_0 = -1,\ w_1 = 1.1,\ w_2 = 1.1$$



XOR?

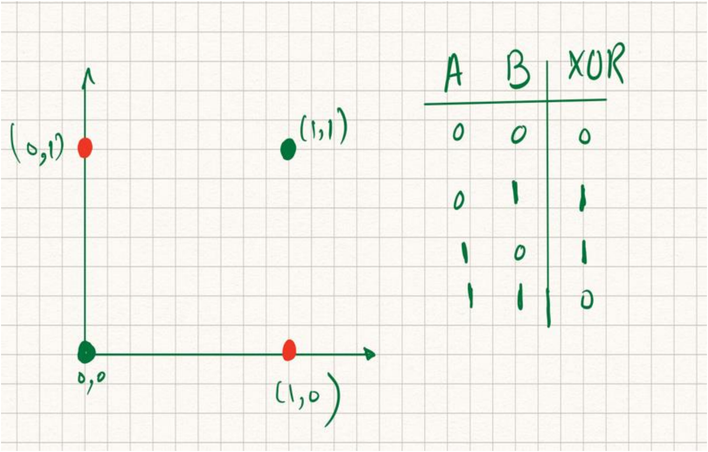


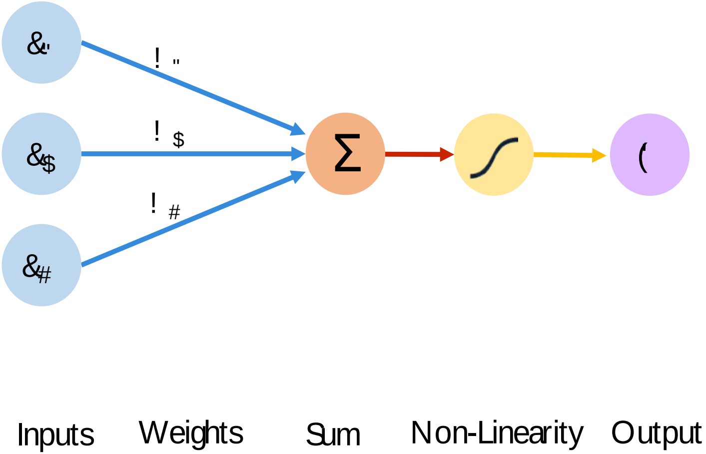


Minsky and Paport, 1968

Carnegie Mellon University

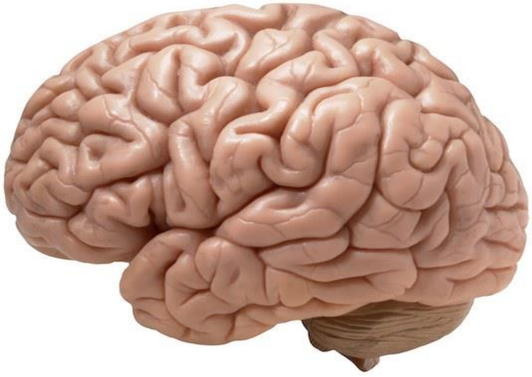
Perceptron can't do XOR

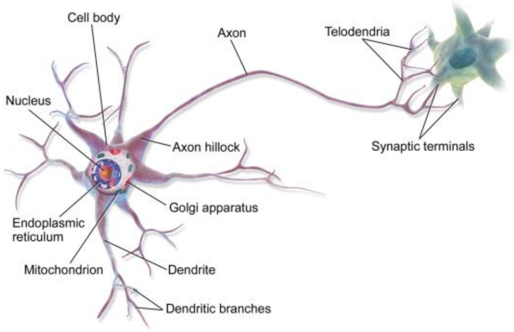


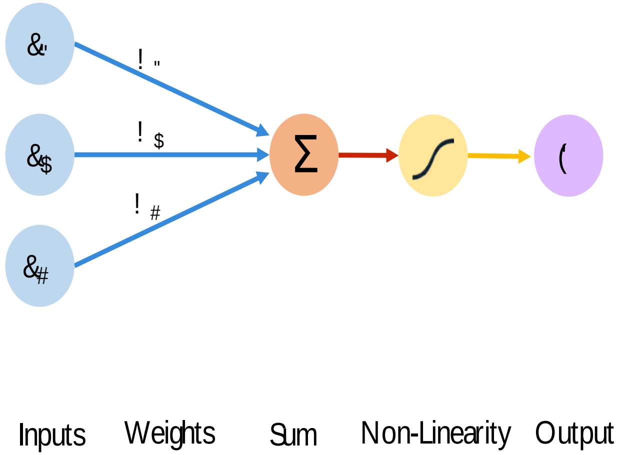


Minsky and Papert in 1969

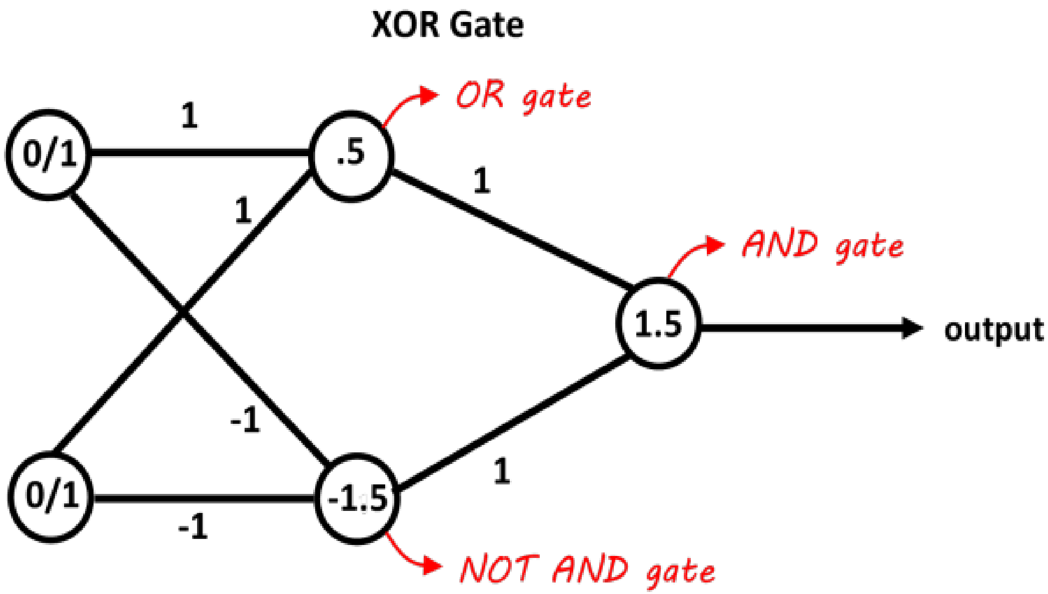
Single Neuron is not ENOUGH

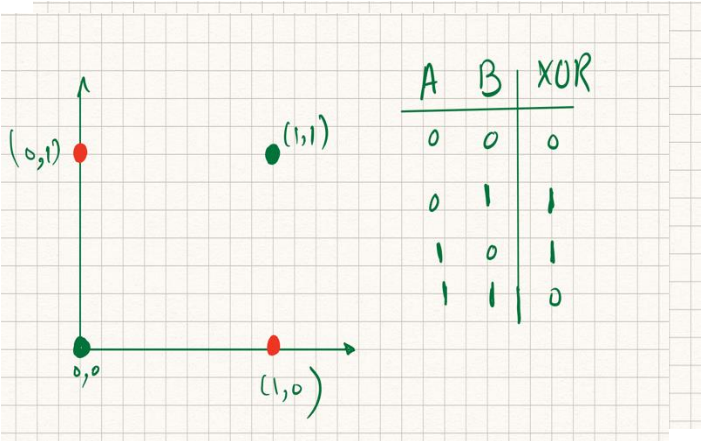




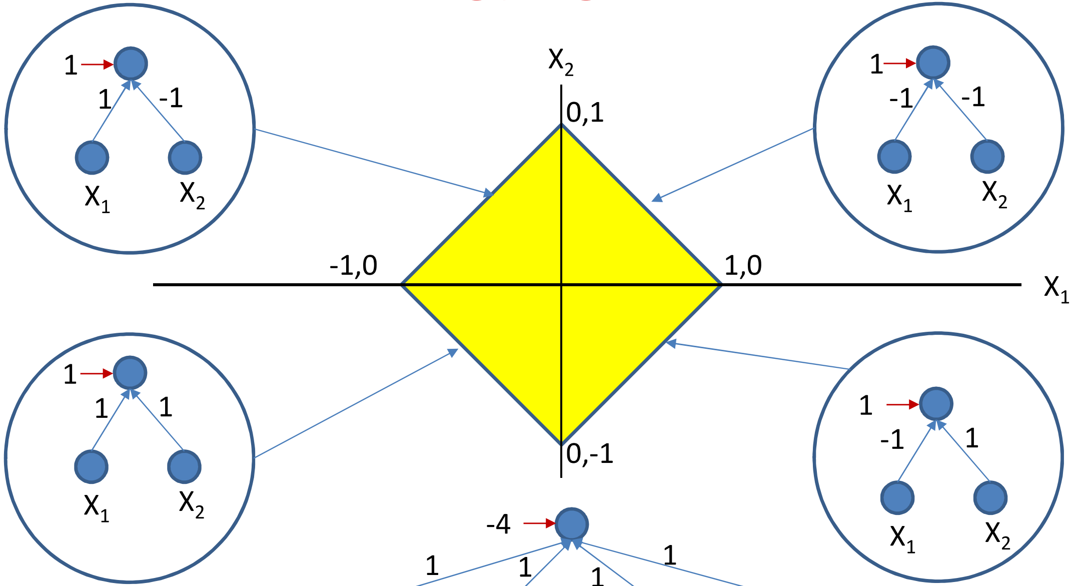


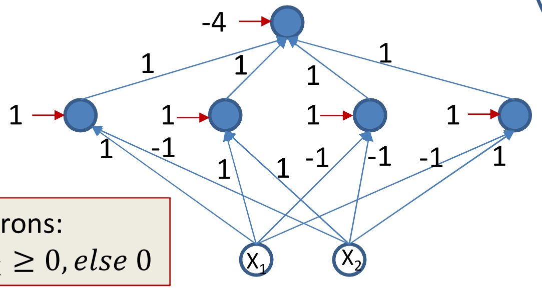
Multilayer Perceptron (MLP) can do XOR





Combining Gates to build MLP by hand



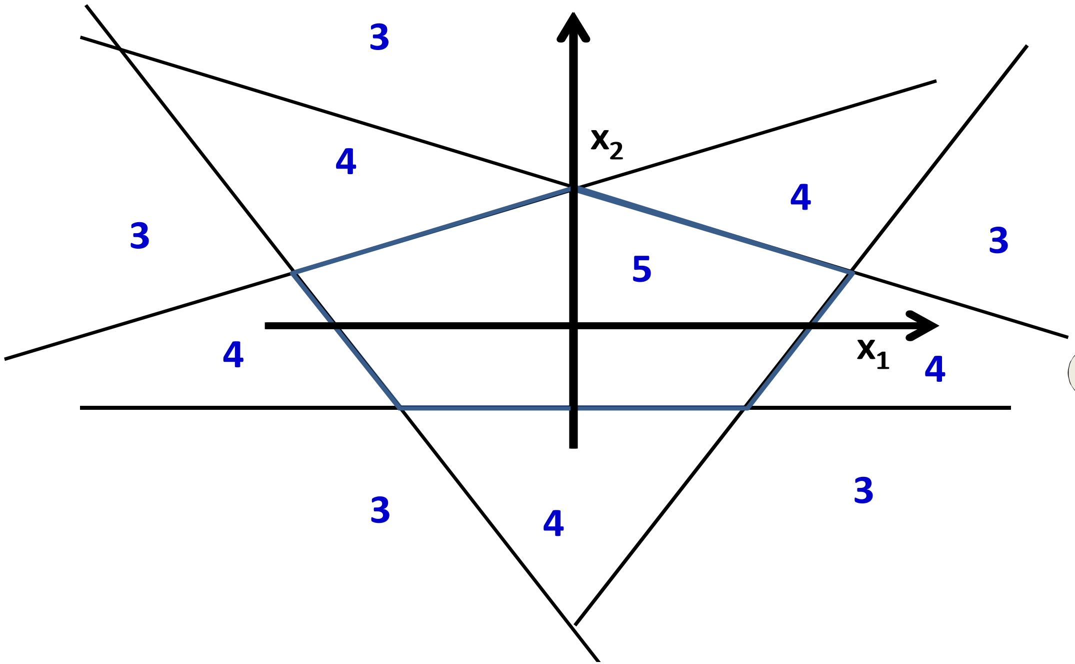


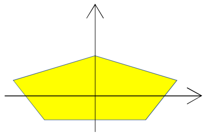
Assuming simple perceptrons:
output = 1 if $\sum_i w_i x_i + b_i \geq 0, else\ 0$

Source: 11785 lecture notes

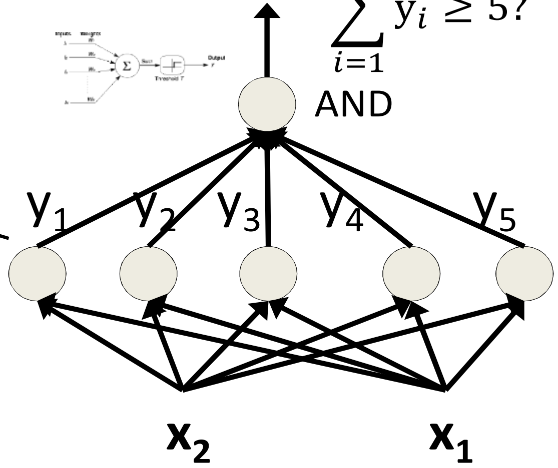
Carnegie Mellon University

More complex Decision Boundaries



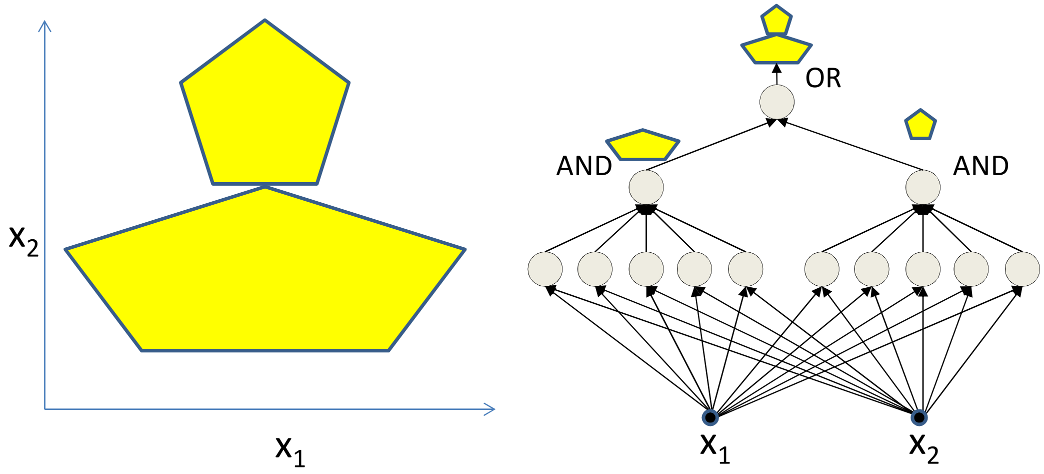


$$\sum_{i=1}^{N} y_i \geq 5?$$



Source: 11785 lecture notes

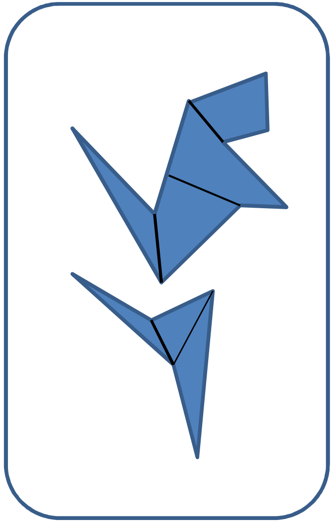
More complex Decision Boundaries

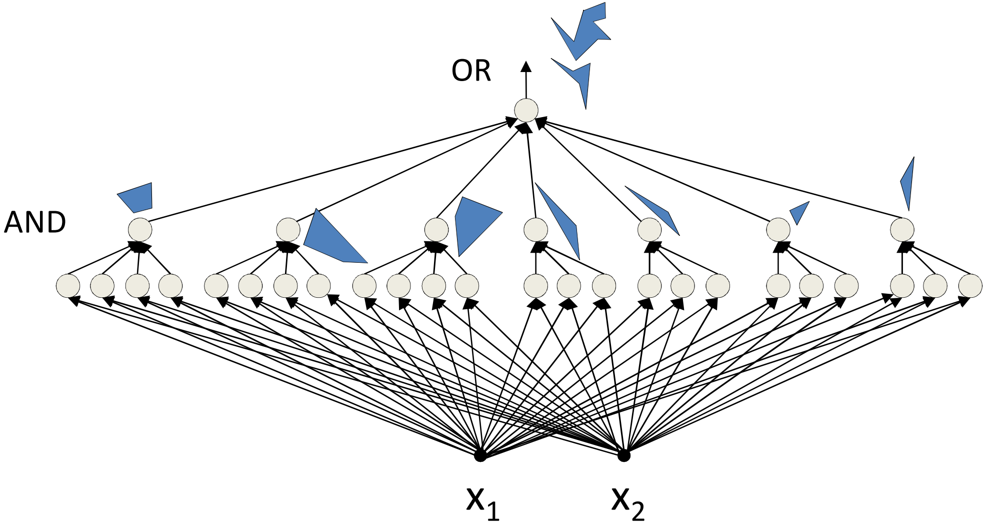


- Network to fire if the input is in the yellow area
  - "OR" two polygons
  - A third layer is required

Source: 11785 lecture notes

More complex Decision Boundaries

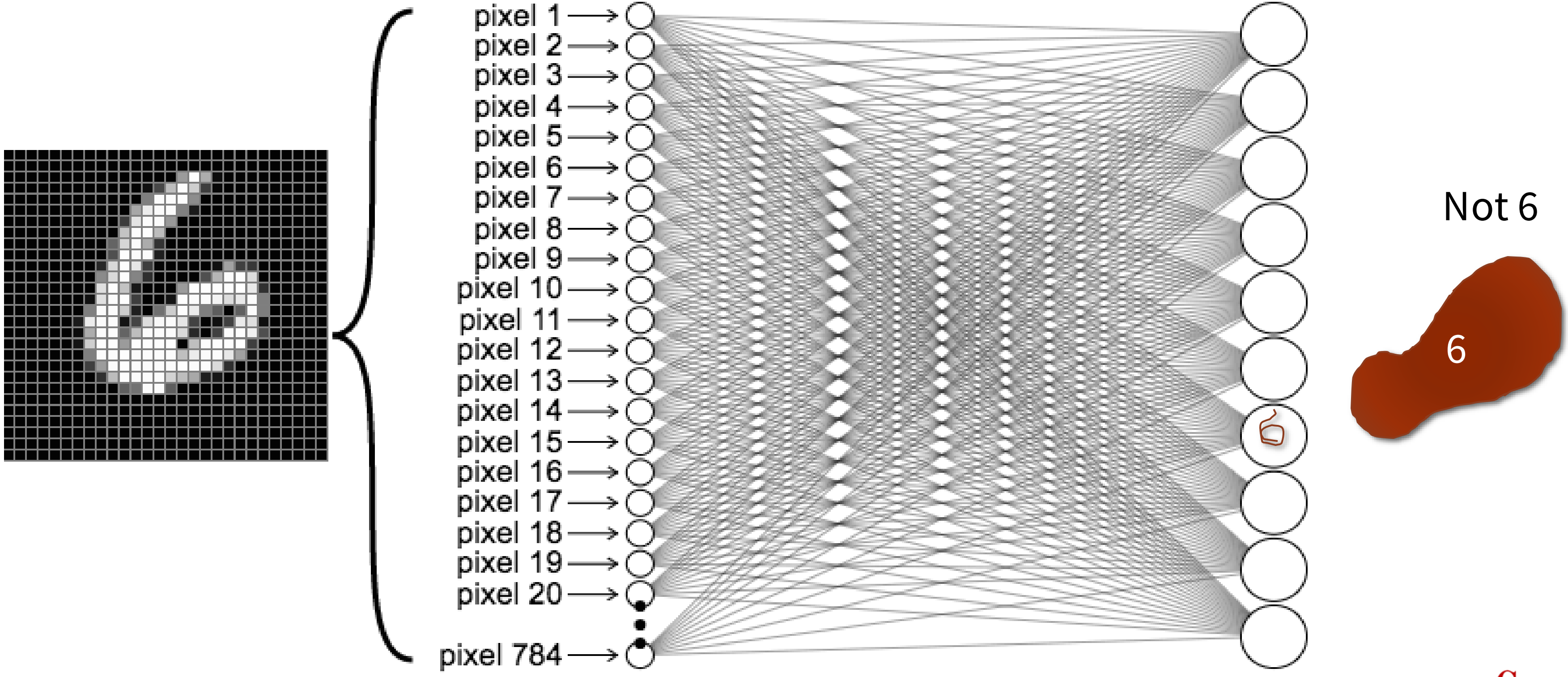




- Can compose *arbitrarily* complex decision boundaries

Source: 11785 lecture notes

Digit Recognition



Course Topics

- Introduction to Deep Learning and its application
- Neural Networks
- Convolutional Neural Networks (CNN)
- Training and Testing CNN
- Interpretability of Deep Learning
- Solving Engineering problems using Deep Learning

Pytorch

```
# loading PyTorch
import torch

# cuda
import torch.cuda as tCuda # various functions and settings
torch.backends.cudnn.deterministic = True # deterministic ML?
torch.backends.cudnn.benchmark = False  # deterministic ML?
torch.cuda.is_available # check if cuda is is_available
tensor.cuda() # moving tensor to gpu
tensor.cpu() # moving tensor to cpu
tensor.to(device) # copy densor to device xyz
torch.device('cuda') # or 'cuda0', 'cuda1' if multiple devices
torch.device('cpu') # default

# static computation graph/C++ export preparation
torch.jit.trace()
from torch.jit import script, trace
@script

# load and save a model
torch.save(model, 'model_file')
model = torch.load('model_file')
model.eval() # set to inference
torch.save(model.state_dict(), 'model_file') # only state dict
model = ModelCalss()
model.load_state_dict(torch.load('model_file'))
```

Carnegie Mellon University