MSIS549: ML and AI for Business Applications
Lesson 1

Syllabus

Location	Zoom
Instructor TA	Jun Yu (jnyu@uw.edu) Xiyuan Ge (xiyuange@uw.edu)
Canvas	https://canvas.uw.edu/courses/1315587
Textbook	• Deep Learning with Python by François Chollet • Deep Learning by Aaron C. Courville, Ian Goodfellow, and Yoshua Bengio
Software	Python & Google Colab
Grading	• 3 Assignments (90%) [20% penalty applied every 24 hrs] • Quiz & Participation (10%)

Course Outline

Lesson 1: Introduction to Deep Learning

Lesson 2: Neural Network Fundamentals

Lesson 3: Convolutional Neural Networks

Lesson 4: Recurrent Neural Networks

Lesson 5: Big Data

The Rise of Deep Learning

Foster School of Business

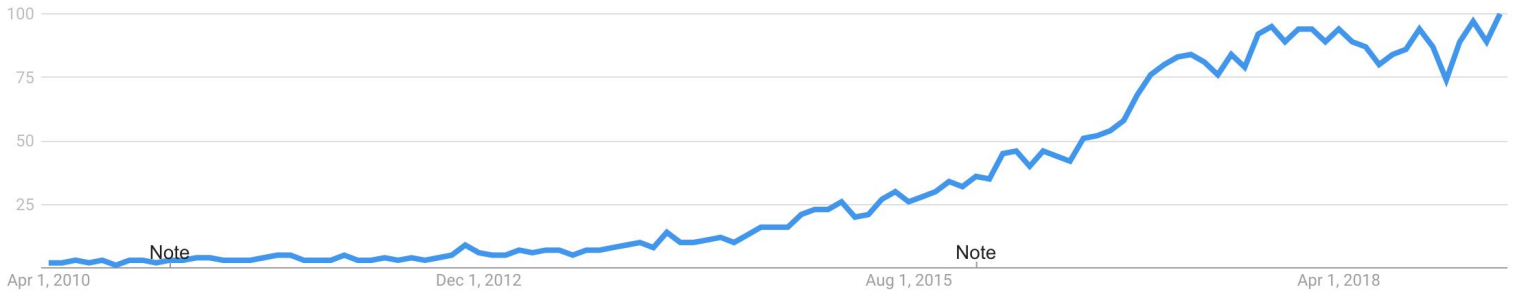
The Rise of Deep Learning

deep learning
Search term

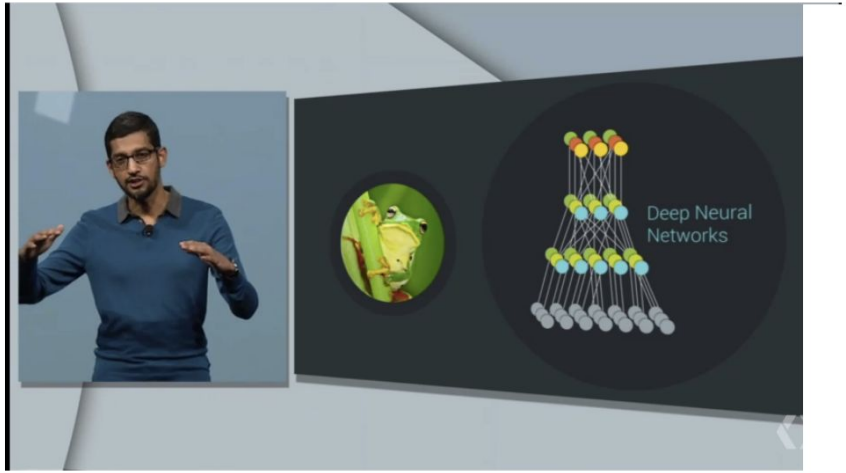
+ Compare

United States | 3/28/10 - 4/28/19 | All categories | Web Search

Interest over time



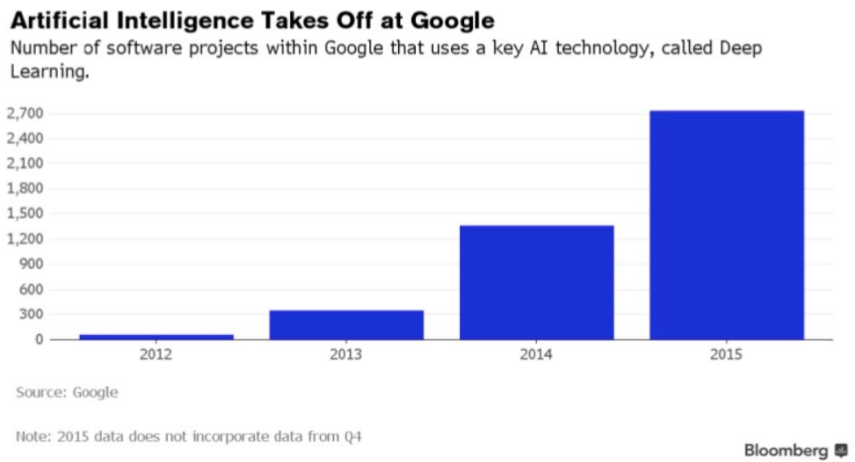
The Rise of Deep Learning



"We will move
from a mobile-first world to
an AI-first world"

- Sundar Pichai, Google.
Letter to shareholders

Google: Hype or Reality?



Companies and startups use DL

Google amazon.com

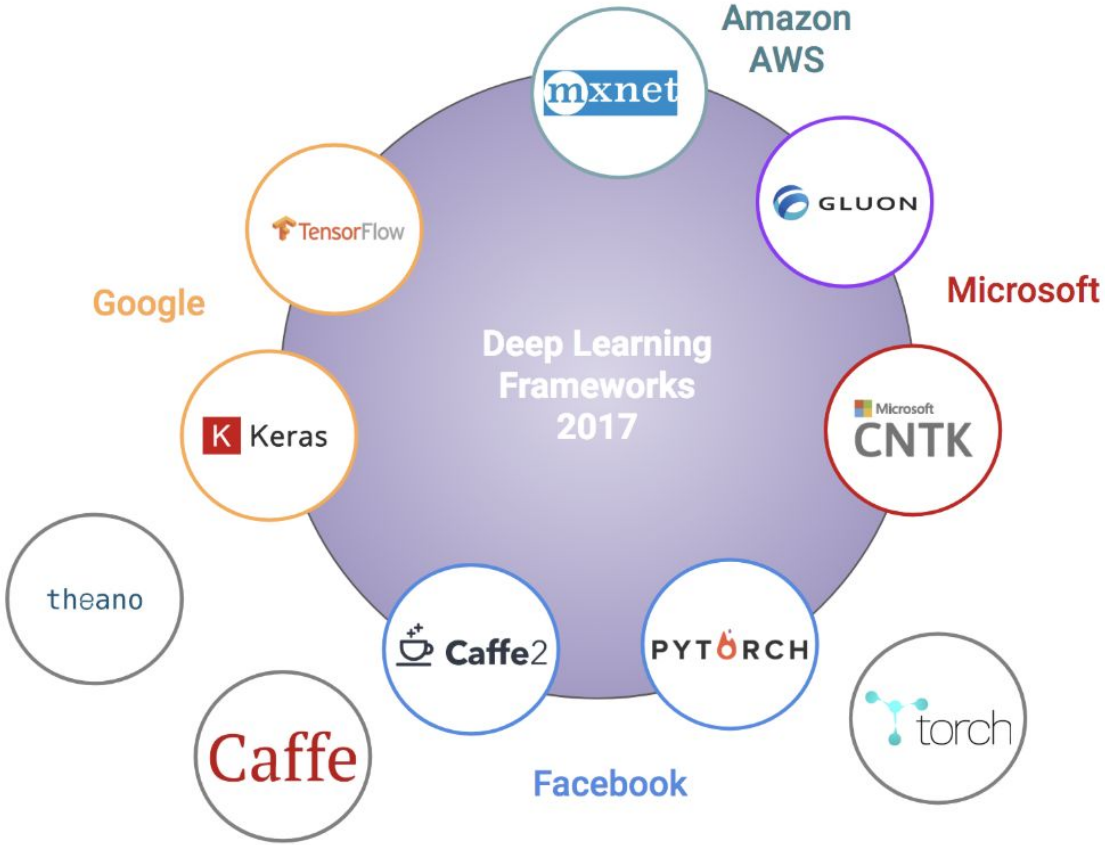
facebook

Microsoft

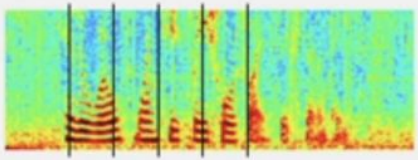
Baidu 百度

TESLA

DL Frameworks

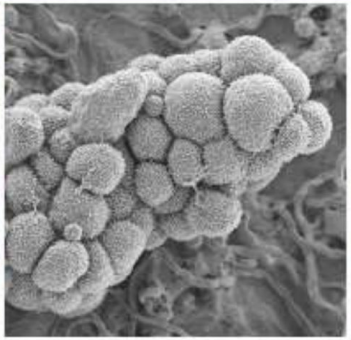
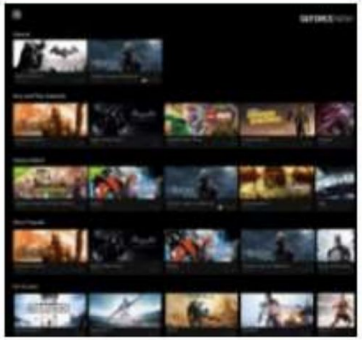


Deep Learning Applications

input		output
Pixels:		"lion"
Audio:		"How cold is it outside?"
"Hello, how are you?"		"Bonjour, comment allez-vous?"
Pixels:		"A blue and yellow train travelling down the tracks"

Can you think of any DL applications?

Deep Learning Everywhere

INTERNET & CLOUD

Image Classification
Speech Recognition
Language Translation
Language Processing
Sentiment Analysis
Recommendation

MEDICINE & BIOLOGY

Cancer Cell Detection
Diabetic Grading
Drug Discovery

MEDIA & ENTERTAINMENT

Video Captioning
Video Search
Real Time Translation

SECURITY & DEFENSE

Face Detection
Video Surveillance
Satellite Imagery

AUTONOMOUS MACHINES

Pedestrian Detection
Lane Tracking
Recognize Traffic Sign

Errors in Image Captioning

"a young boy is holding a baseball bat."

What's common in the images?

https://thispersondoesnotexist.com/

Foster School of Business

Deep Learning, ML and AI

Artificial Intelligence

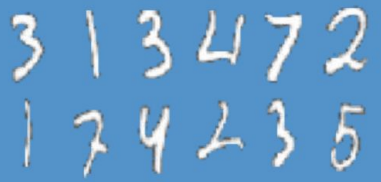
Any technique that enables computers to mimic human behavior

Machine Learning

Ability to learn without explicitly being programmed

Deep Learning

Extract patterns from data using neural networks



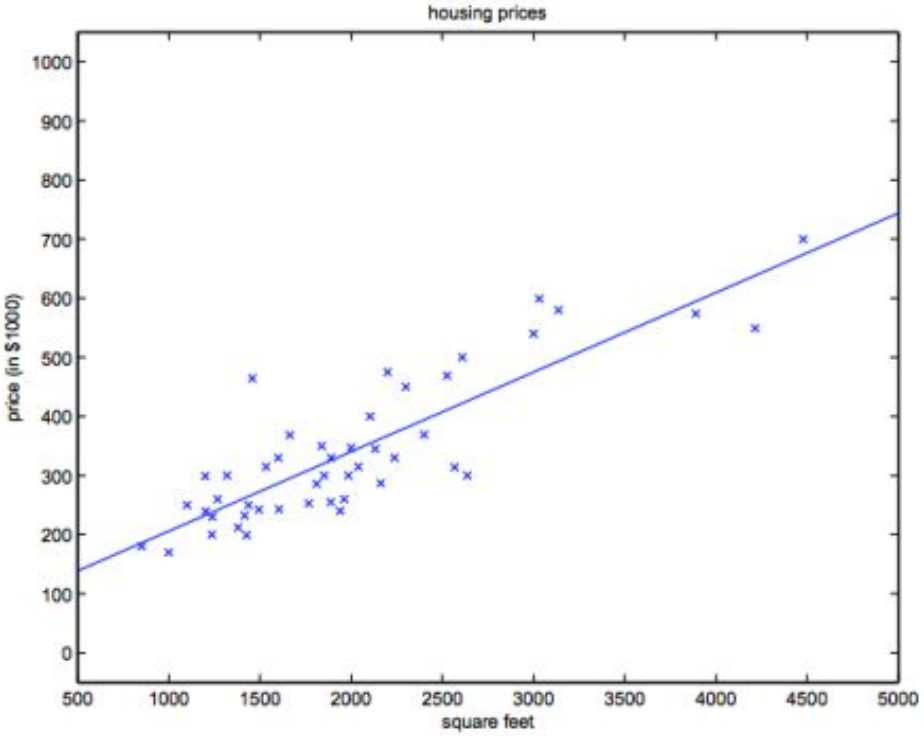
Machine Learning

- MACHINE LEARNING is a type of ARTIFICIAL INTELLIGENCE that provides computers the ability to learn without being explicitly programmed.
- Approximate a mapping function from input X to output Y, based on some sample data.

Problem	X	Y
Housing Price Prediction	Size (sq. ft), location	$35,000
Spam Detection	Email	Spam / Not Spam
Product recommendations	Product and user features	P(purchase)
Loan approval	Loan application	Will they pay? (0 or 1)
Preventive maintenance	Sensors from planes / hard disk	Is it about to fail?

Machine Learning Example

Housing Price Prediction.



features output

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

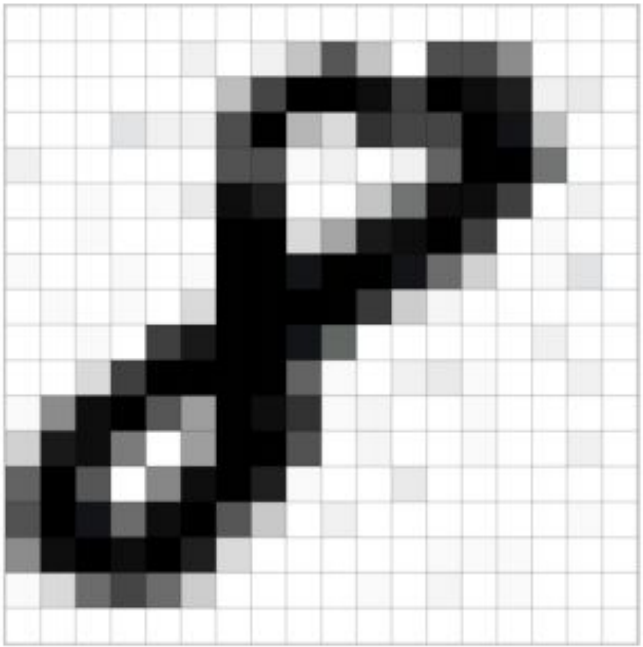
http://cs229.stanford.edu/notes/cs229-notes1.pdf

Perceptual Problems are Challenging

- Traditional ML are not good at perceptual tasks that involving skills that seem natural and intuitive to humans.

- For example:
  - Image classification
  - Speech recognition
  - Learning a new language
  - Driving cars
  - Etc.

What we see vs. What computers see



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0
0	0	1	0	0	0	41	160	250	255	235	162	255	238	206	11	13	0
0	0	0	16	9	9	150	251	45	21	184	159	154	255	233	40	0	0
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	107	0	0
0	0	3	0	4	15	236	216	0	0	38	109	247	240	169	0	11	0
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0
6	0	0	4	0	3	252	250	228	255	255	234	112	28	0	2	17	0
0	2	1	4	0	21	255	253	251	255	172	31	8	0	1	0	0	0
0	0	4	0	163	225	251	255	229	120	0	0	0	0	0	11	0	0
0	0	21	162	255	255	254	255	126	6	0	10	14	6	0	0	9	0
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	11	0
125	255	141	0	87	244	255	208	3	0	0	13	0	1	0	1	0	0
145	248	228	116	235	255	141	34	0	11	0	1	0	0	0	1	3	0
85	237	253	246	255	210	21	1	0	1	0	0	6	2	4	0	0	0
6	23	112	157	114	32	0	0	0	0	2	0	8	0	7	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

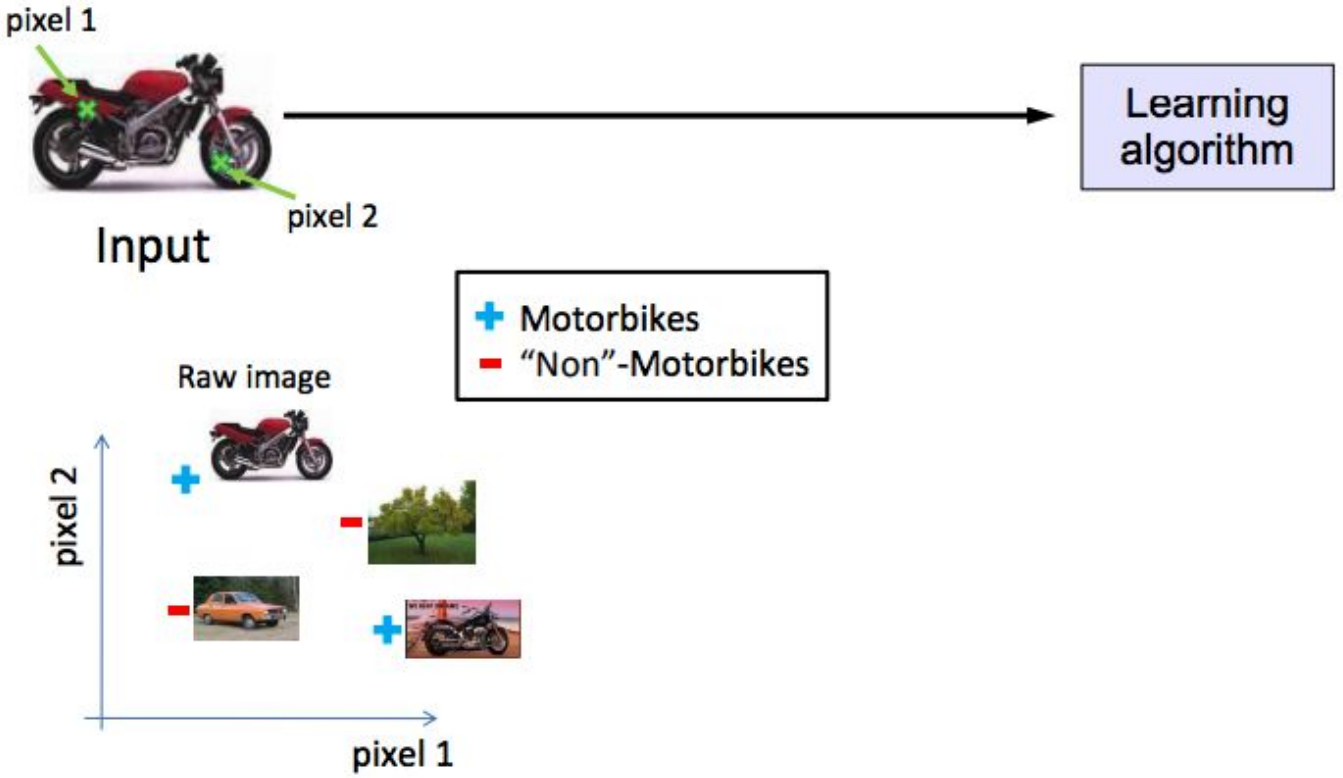
Why it's hard for ML at perceptual problems?

You see this:

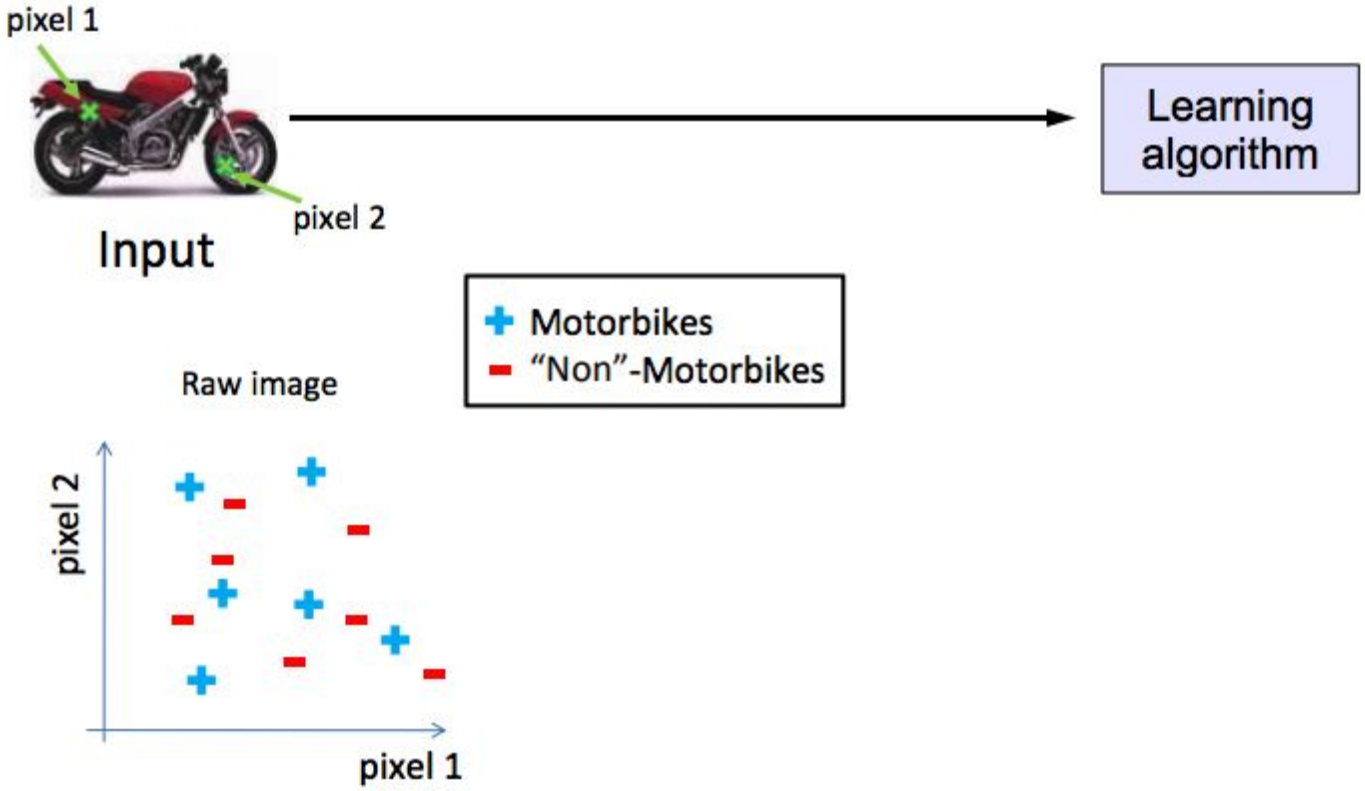
But the camera sees this:

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

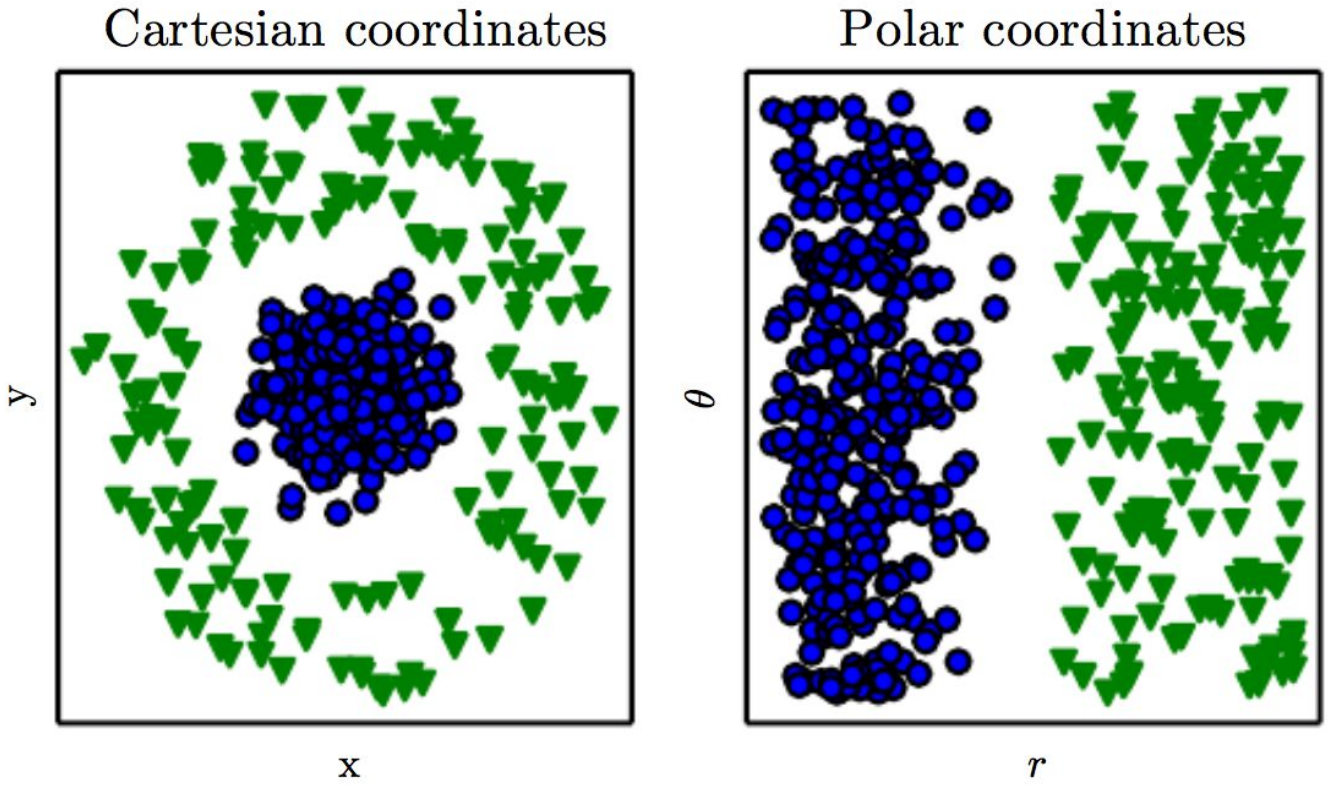
Why it's hard for ML at perceptual problems?



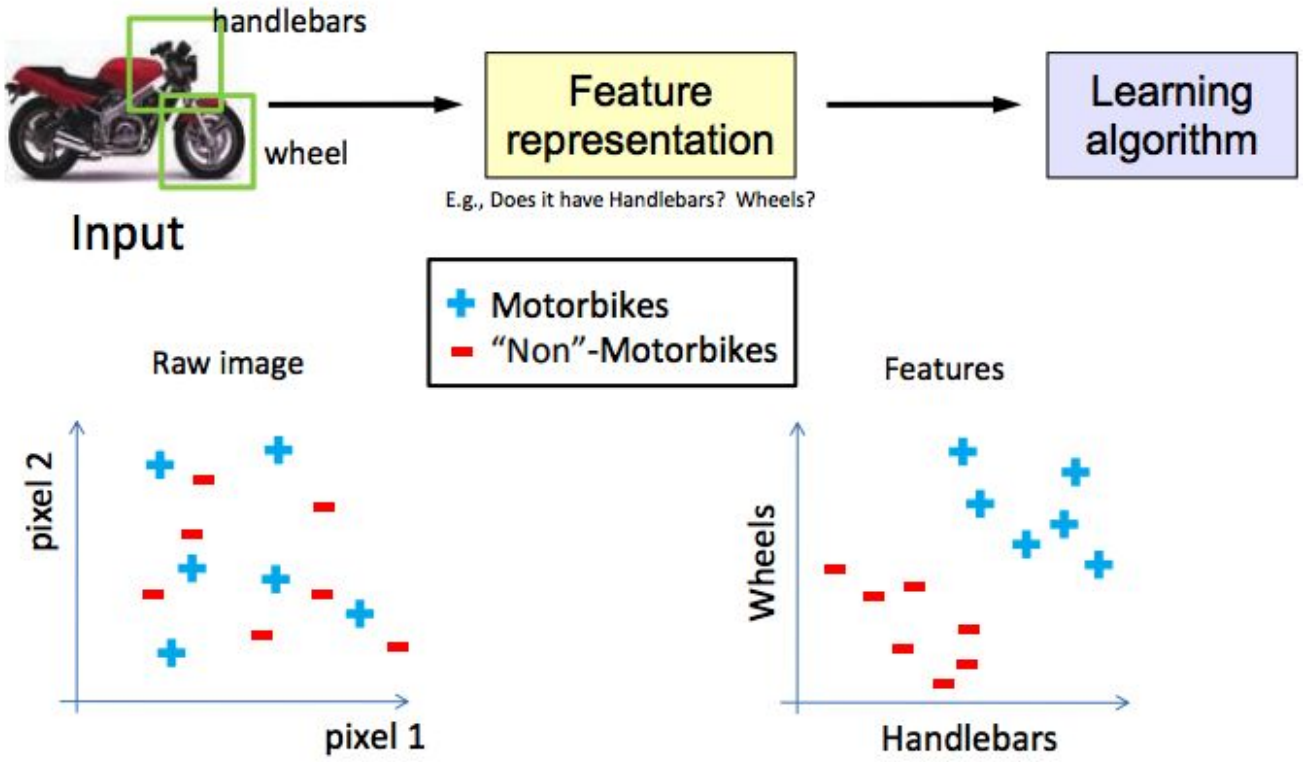
Why it's hard for ML at perceptual problems?



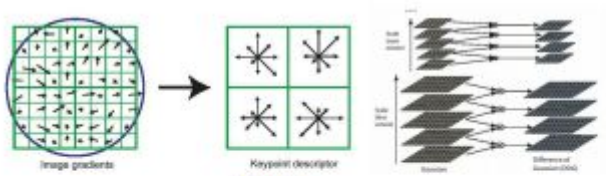
Feature Representation Matters A LOT



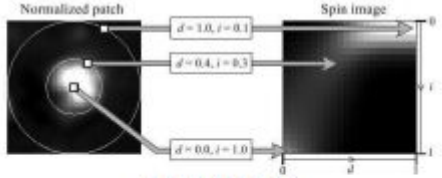
Why is ML bad at perceptual problems?



Feature representations in Computer Vision

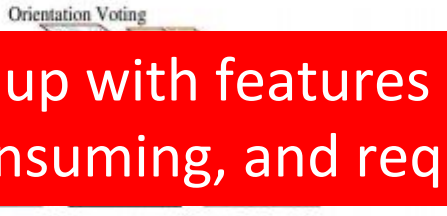


SIFT



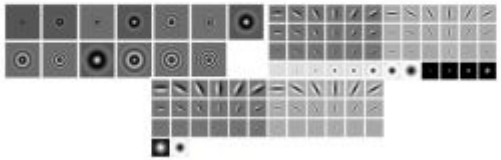
Spin image

Coming up with features is often difficult, time-consuming, and requires expert knowledge.

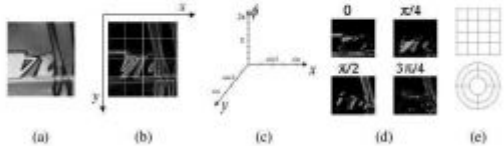


HoG

RIFT



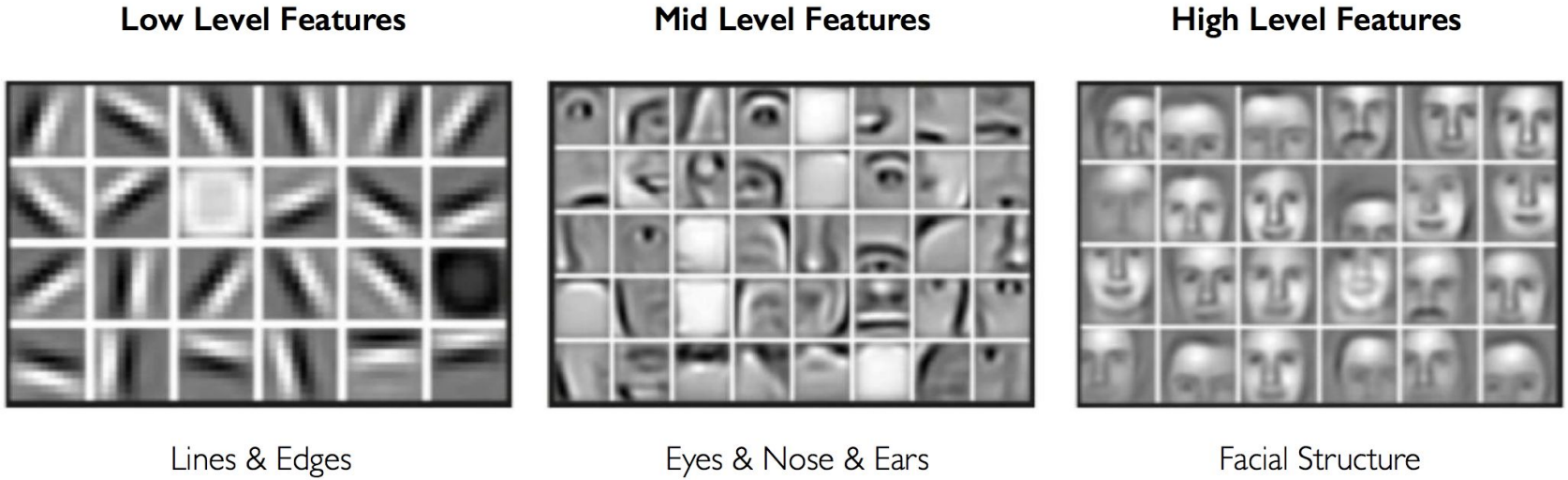
Textons



GLOH

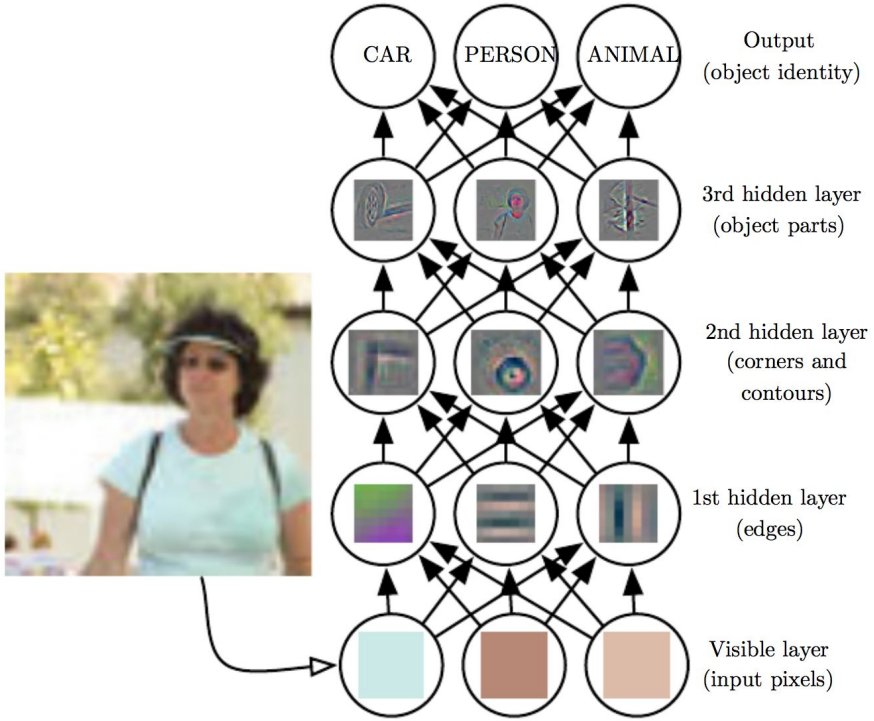
Can we learn the representation?

Hand engineered features are time consuming, brittle and not scalable in practice. Can we learn the underlying features directly from data?

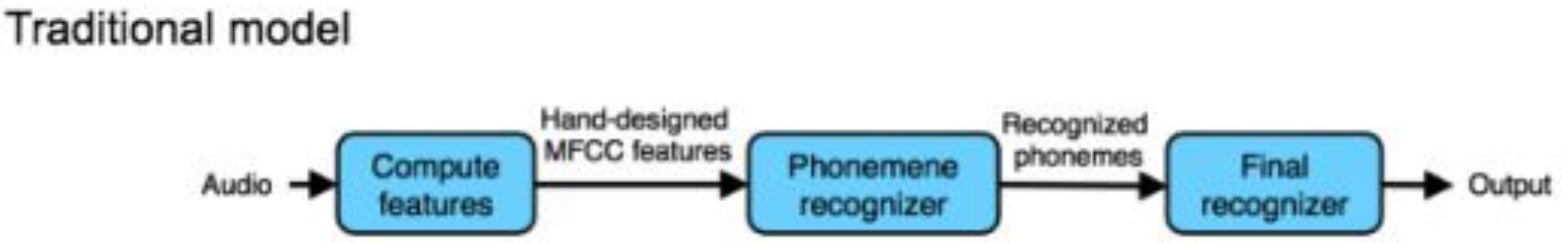


Learn The Representation

Build high-level features from low-level features.



Speech Recognition



- Feature Engineering
  - Time-consuming
  - Domain and task-specific
  - Requires human experts

Does this contains a human face?


```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 12, 0, 11, 39, 137, 37, 0, 152, 147, 84, 0, 0, 0, 0,
0, 1, 0, 0, 0, 41, 160, 250, 255, 235, 162, 255, 238, 206, 11, 13, 0, 0, 0, 0, 16, 9, 9, 150, 251, 45, 21, 184, 159, 154, 2
55, 233, 40, 0, 0, 10, 0, 0, 0, 0, 0, 145, 146, 3, 10, 0, 11, 124, 253, 255, 107, 0, 0, 0, 0, 3, 0, 4, 15, 236, 216, 0, 0,
38, 109, 247, 240, 169, 0, 11, 0, 1, 0, 2, 0, 0, 0, 253, 253, 23, 62, 224, 241, 255, 164, 0, 5, 0, 0, 6, 0, 0, 4, 0, 3, 252
, 250, 228, 255, 255, 234, 112, 28, 0, 2, 17, 0, 0, 2, 1, 4, 0, 21, 255, 253, 251, 255, 172, 31, 8, 0, 1, 0, 0, 0, 0, 0, 4,
 0, 163, 225, 251, 255, 229, 120, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0, 21, 162, 255, 255, 254, 255, 126, 6, 0, 10, 14, 6, 0, 0, 9
, 0, 3, 79, 242, 255, 141, 66, 255, 245, 189, 7, 8, 0, 0, 5, 0, 0, 0, 0, 26, 221, 237, 98, 0, 67, 251, 255, 144, 0, 8, 0, 0
, 7, 0, 0, 11, 0, 125, 255, 141, 0, 87, 244, 255, 208, 3, 0, 0, 13, 0, 1, 0, 1, 0, 0, 145, 248, 228, 116, 235, 255, 141, 34
, 0, 11, 0, 1, 0, 0, 0, 1, 3, 0, 85, 237, 253, 246, 255, 210, 21, 1, 0, 1, 0, 0, 6, 2, 4, 0, 0, 0, 6, 23, 112, 157, 114, 32
, 0, 0, 0, 0, 2, 0, 8, 0, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

End-to-end learning

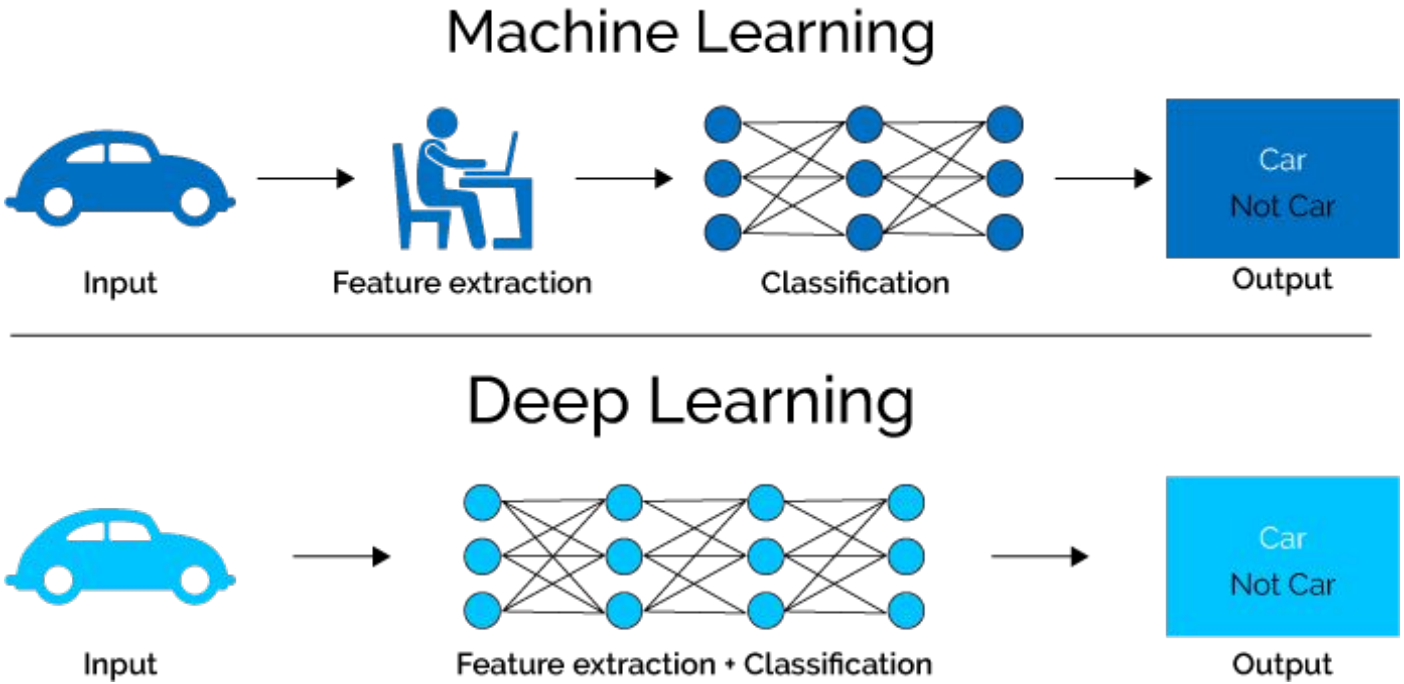


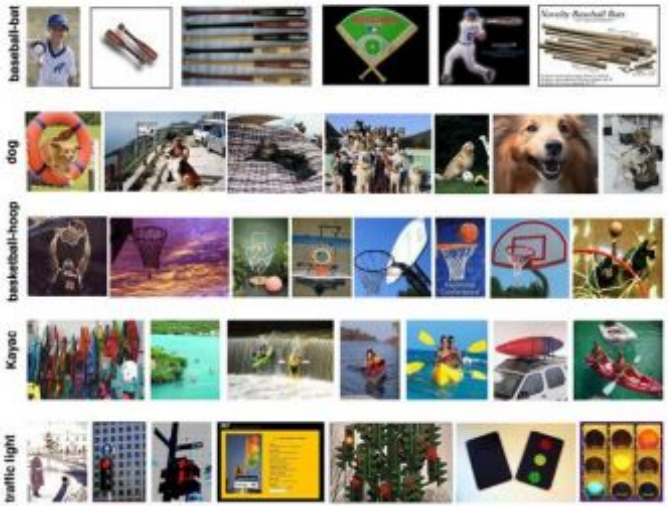
Image Classification: MNIST Data

- MNIST database
  - 60,000 training, 10,000 testing
  - Large enough for digits
  - Battlefield of the 90s

Algorithm	Error Rate (%)
Linear classifier (perceptron)	12.0
K-nearest-neighbors	5.0
Boosting	1.26
SVM	1.4
Neural Network	1.6
Convolutional Neural Networks	0.95
With automatic distortions + ensemble + many tricks	0.23

Image Classification: Caltech-101 Data

- Caltech-101 dataset
  - Around 10,000 images
  - Certainly not enough!



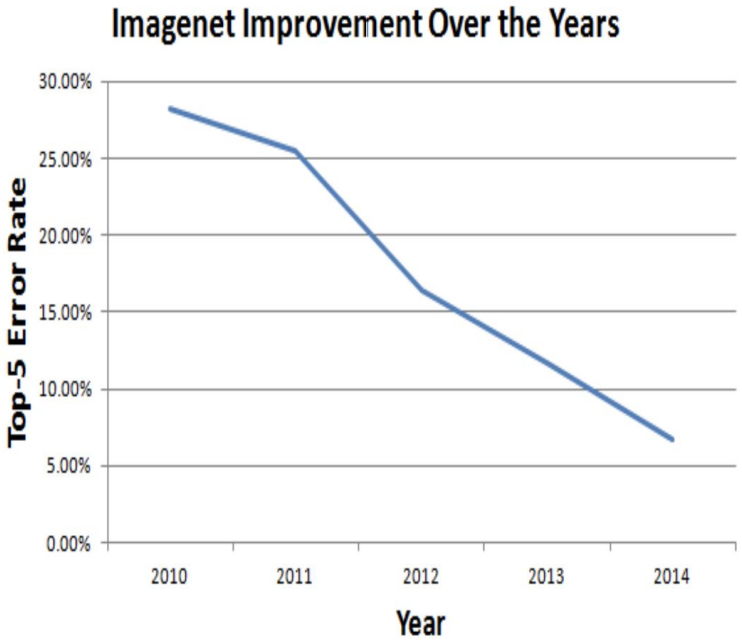
~80% is widely considered to be the limit on this dataset

Algorithm	Accuracy (%)
SVM with Pyramid Matching Kernel (2005)	58.2%
Spatial Pyramid Matching (2006)	64.6%
SVM-KNN (2006)	66.2%
Sparse Coding + Pyramid Matching (2009)	73.2%
SVM Regression w object proposals (2010)	81.9%
Group-Sensitive MKL (2009)	84.3%
Deep Learning (pretrained on Imagenet) (2014)	91.4%

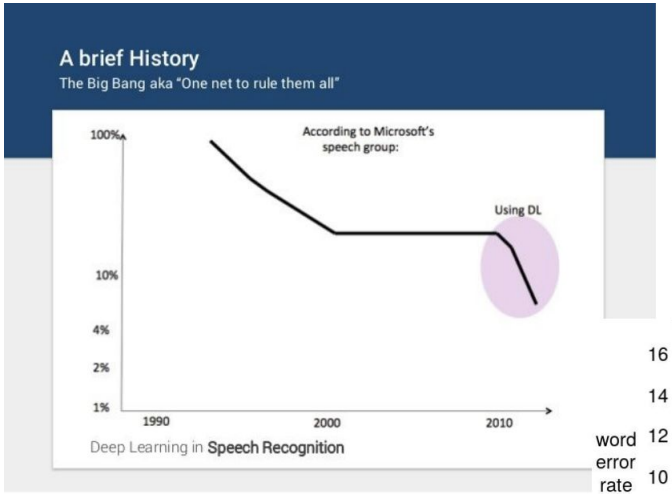
Image Classification: ImageNet Data

- Imagenet
  - Over 1 million images of 1000 classes

- 16.42% Deep CNN dropout in 2012
- 6.66% 22 layer CNN (GoogLeNet) in 2014
- 3.6% (Microsoft Research Asia) super-human performance in 2015

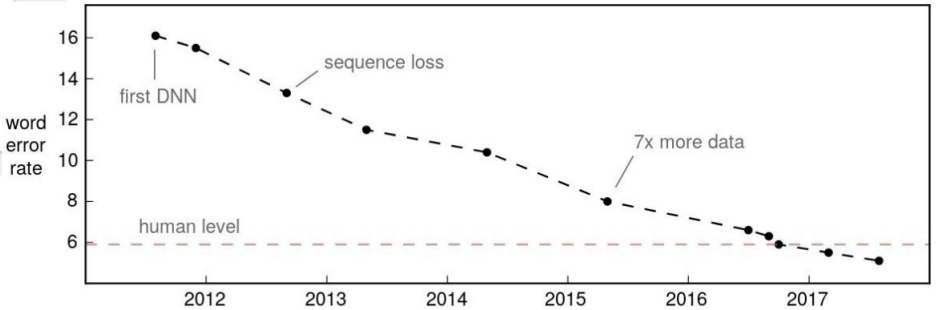


Speech recognition



Source: https://www.slideshare.net/LuMa921/deep-learning-the-past-present-and-future-of-artificial-intelligence

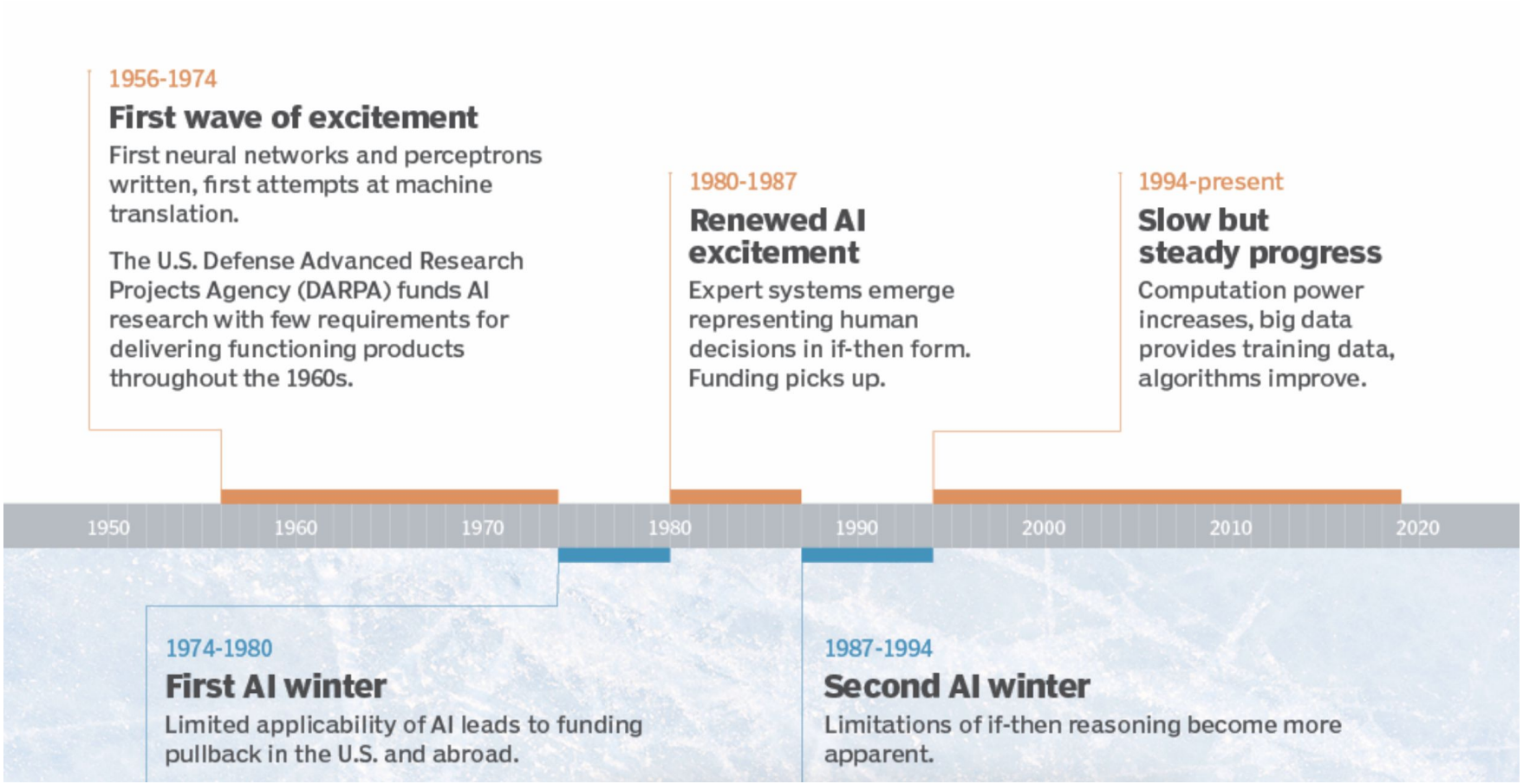
- Significant improvement in Word Error Rate (WER)
  - E.g., A WER of 5% roughly corresponds to 1 missed word for every 20
- Beating traditional approaches and even human?



Improvements in word error rate over time on the Switchboard conversational speech recognition benchmark. The test set was collected in 2000. It consists of 40 phone conversations between two random native English speakers.

Source: https://awni.github.io/speech-recognition/

AI Winters

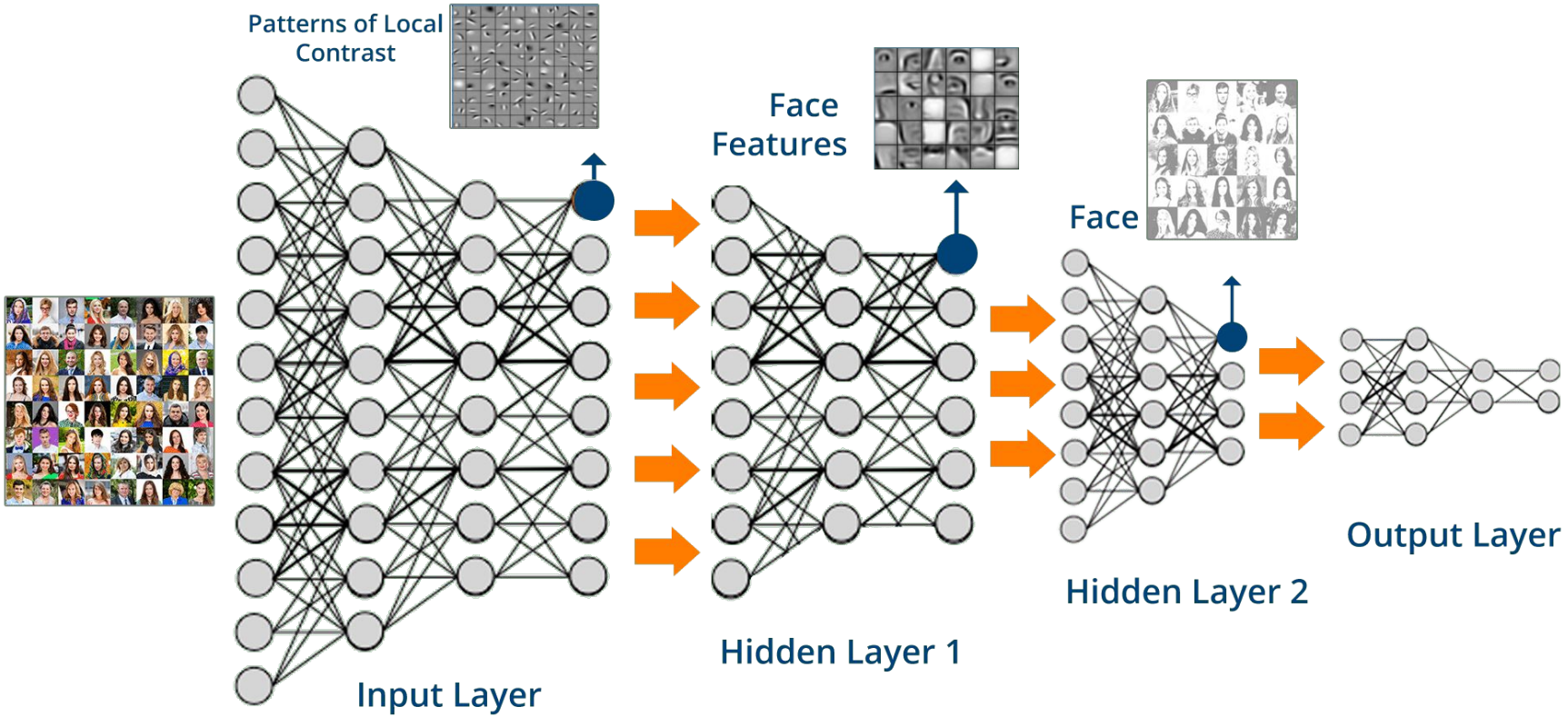

1956-1974
First wave of excitement
First neural networks and perceptrons written, first attempts at machine translation.
The U.S. Defense Advanced Research Projects Agency (DARPA) funds AI research with few requirements for delivering functioning products throughout the 1960s.
1980-1987
Renewed AI excitement
Expert systems emerge representing human decisions in if-then form. Funding picks up.
1994-present
Slow but steady progress
Computation power increases, big data provides training data, algorithms improve.
1950
1960
1970
1980
1990
2000
2010
2020
1974-1980
First AI winter
Limited applicability of AI leads to funding pullback in the U.S. and abroad.
1987-1994
Second AI winter
Limitations of if-then reasoning become more apparent.

What is Deep Learning?

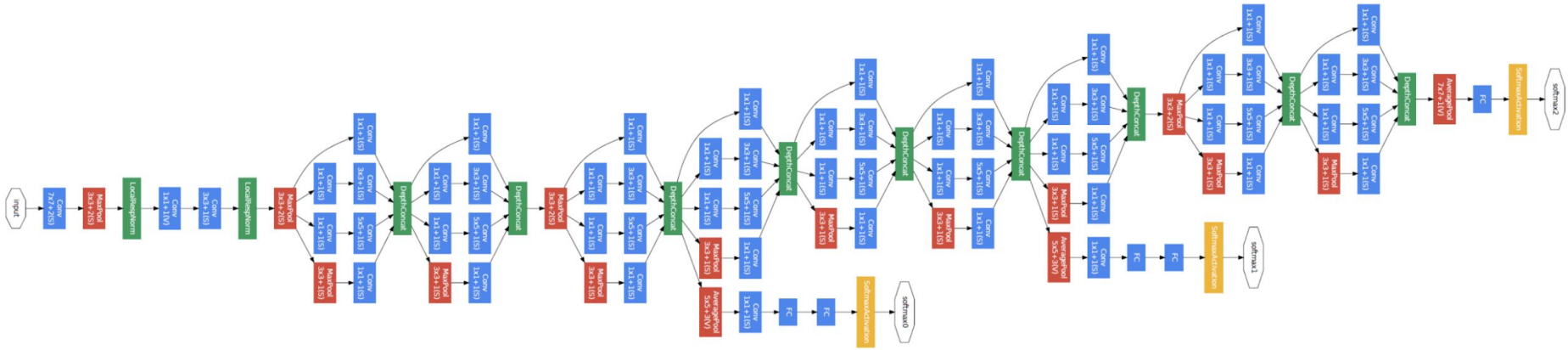
- Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making.

- The ability for computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined in terms of its relation to simpler concepts.

Deep Neural Network



GoogLeNet



GoogLeNet Network (From Left to Right)

Why now?

Neural Networks date back decades, so why the resurgence?

1. Big Data

- Larger Datasets
- Easier Collection & Storage

IMAGENET

WIKIPEDIA
The Free Encyclopedia

2. Hardware

- Graphics Processing Units (GPUs)
- Massively Parallelizable

3. Software

- Improved Techniques
- New Models
- Toolboxes

TensorFlow

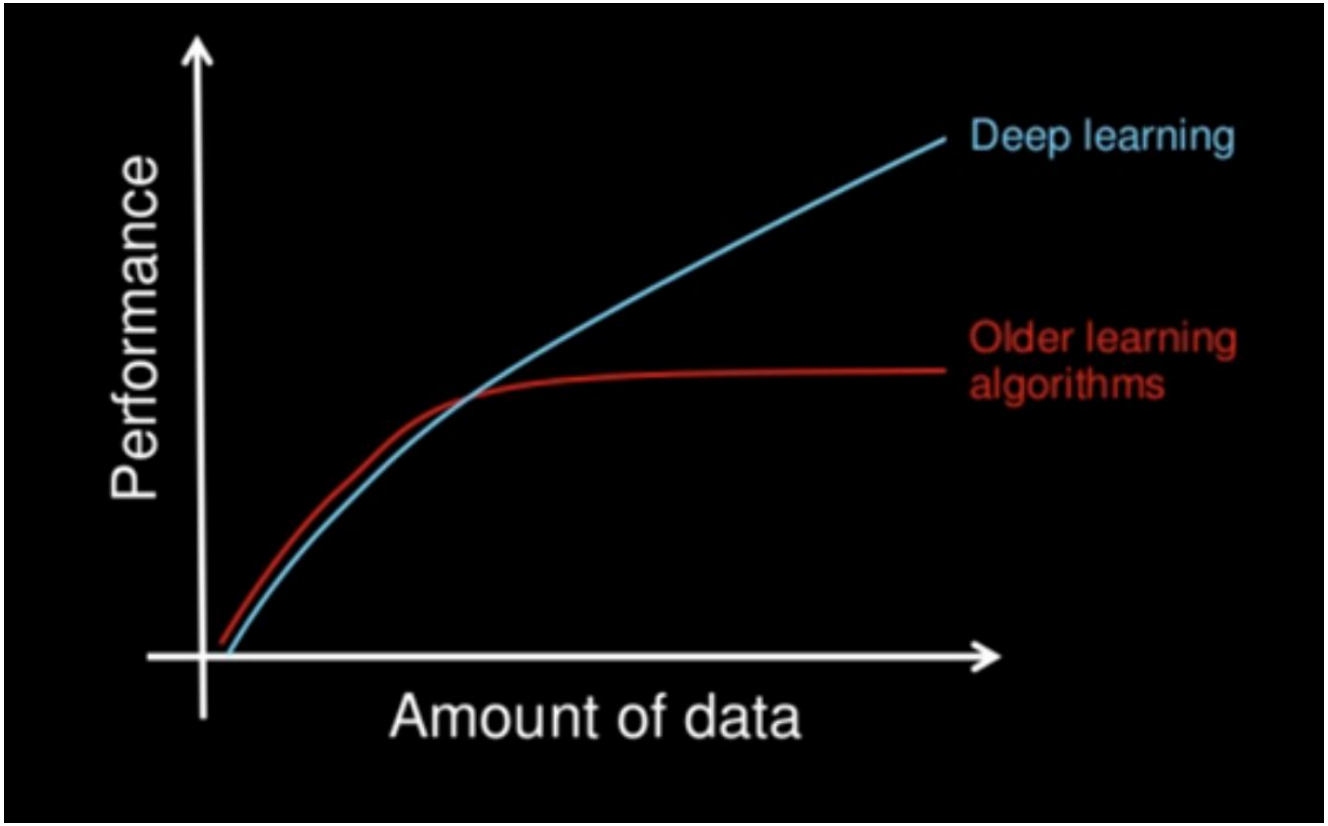
1952 Stochastic Gradient Descent

1958 Perceptron
- Learnable Weights

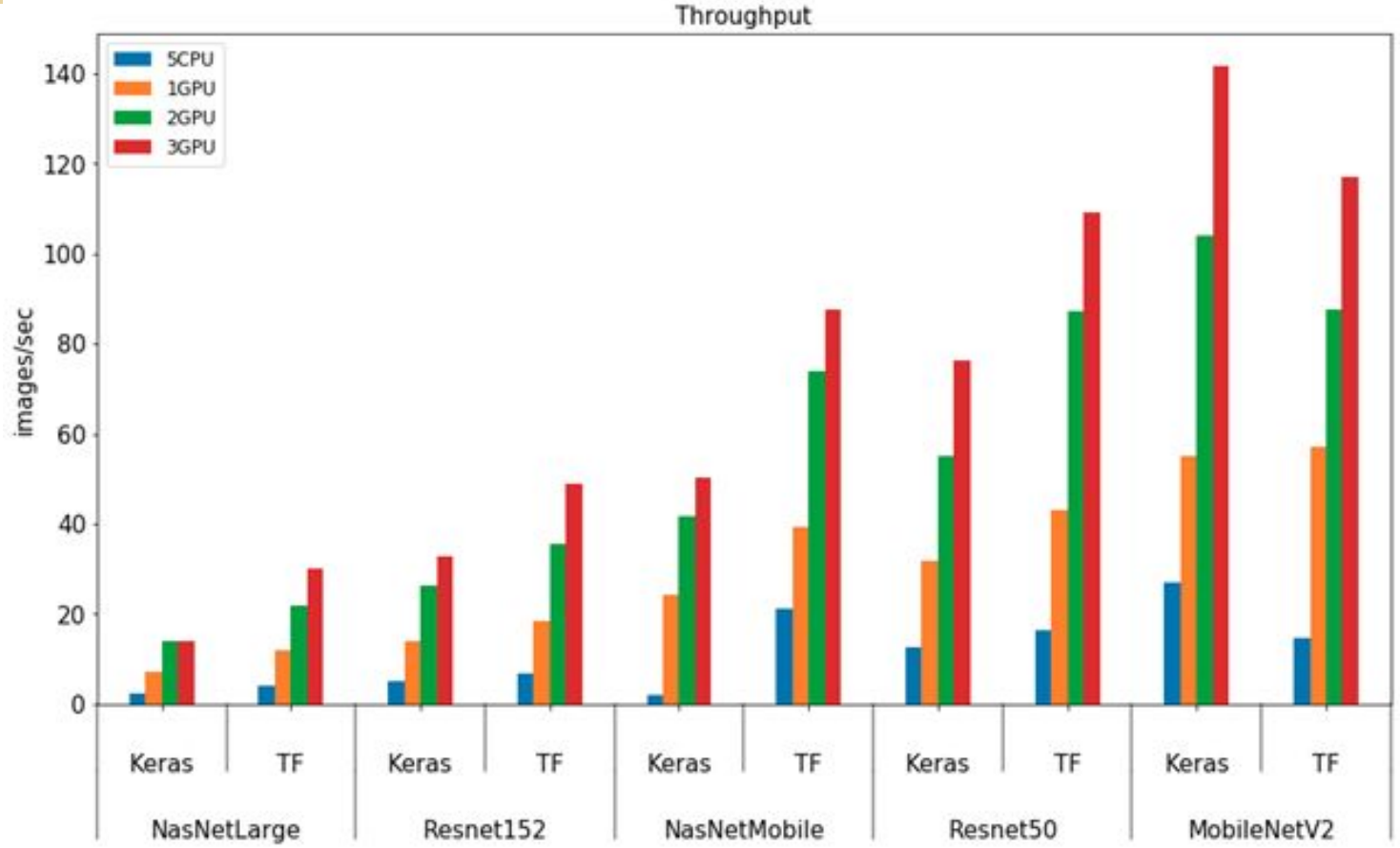
1986 Backpropagation
- Multi-Layer Perceptron

1995 Deep Convolutional NN
- Digit Recognition

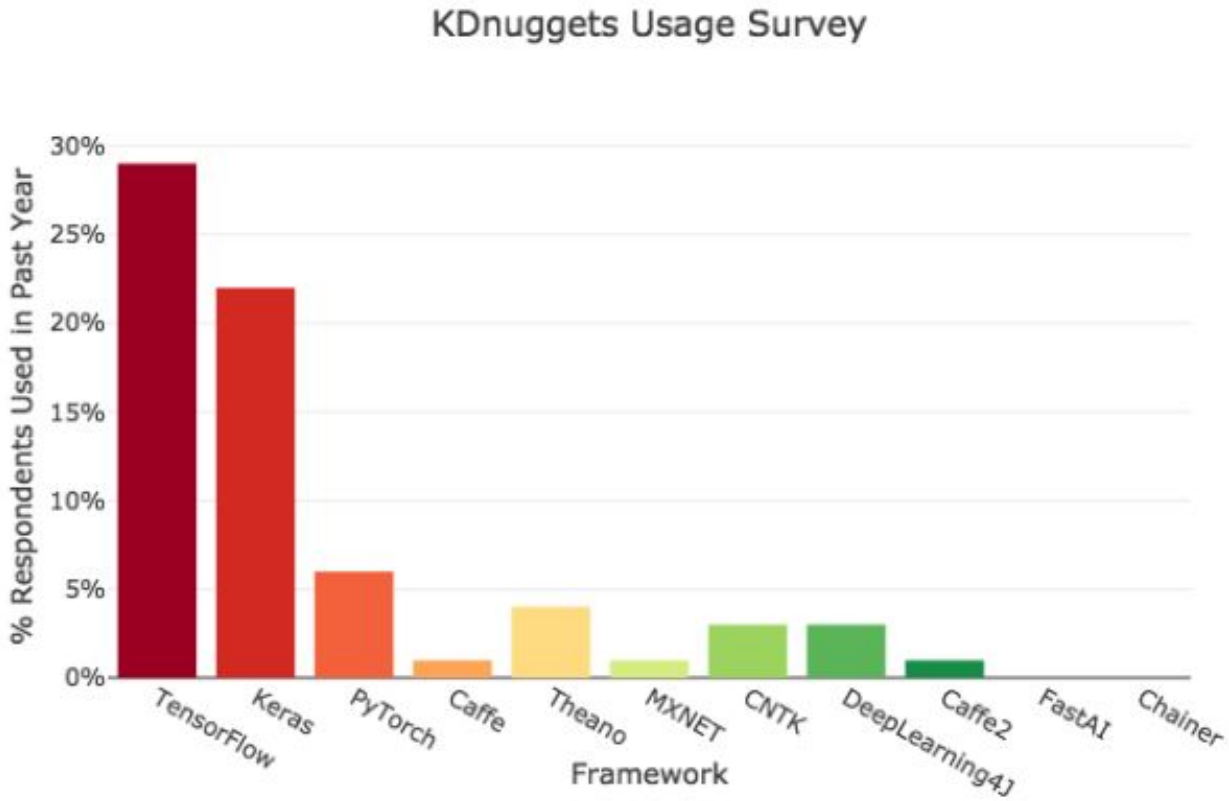
Deep Learning - Big Data



Deep Learning - Hardware



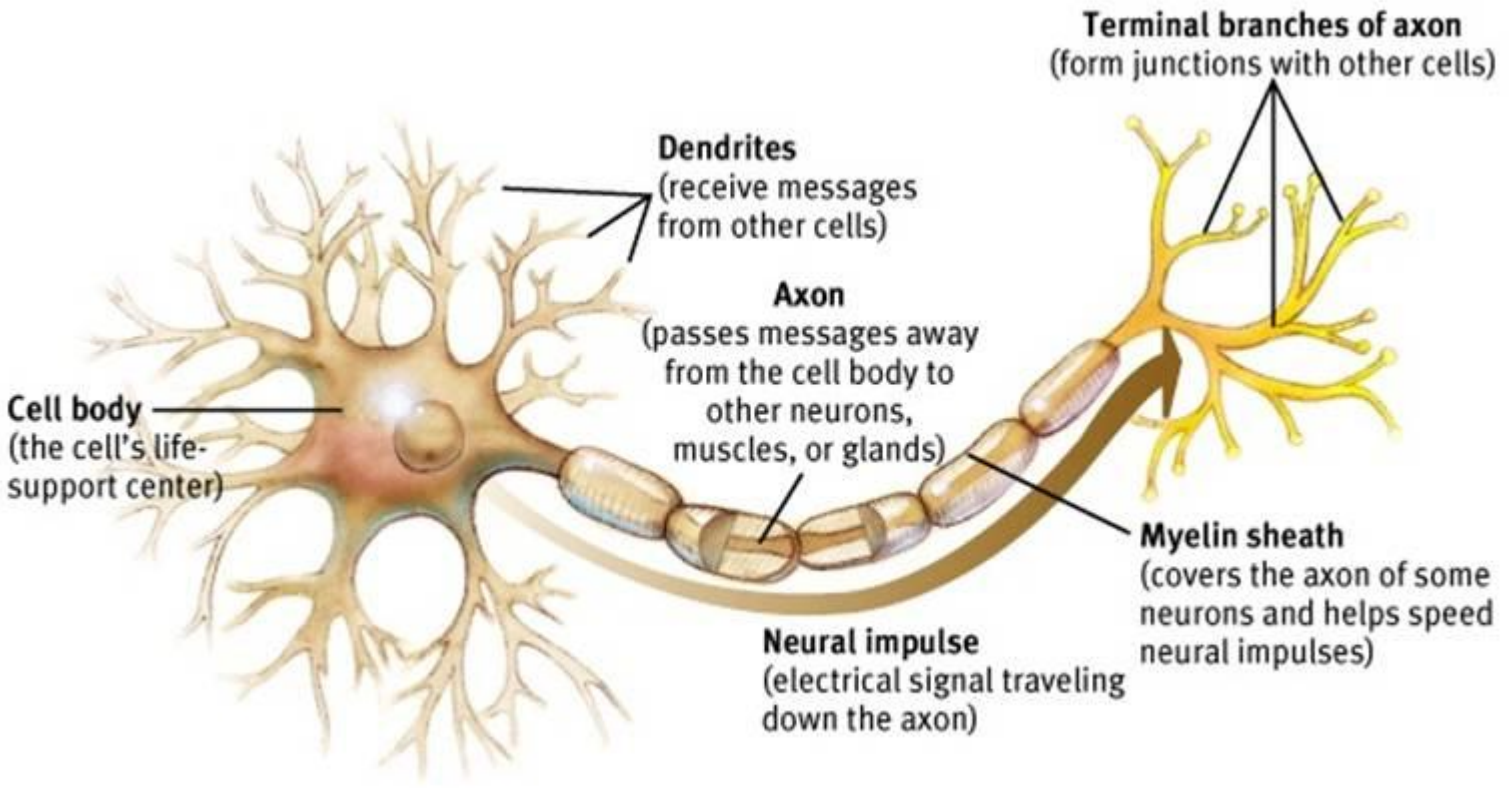
Deep Learning - Software



The Perceptron

The structural building block of deep learning

Introduction and History of Neural Networks



Introduction and History of Neural Networks

> McCollough-Pitts (MCP) neurons

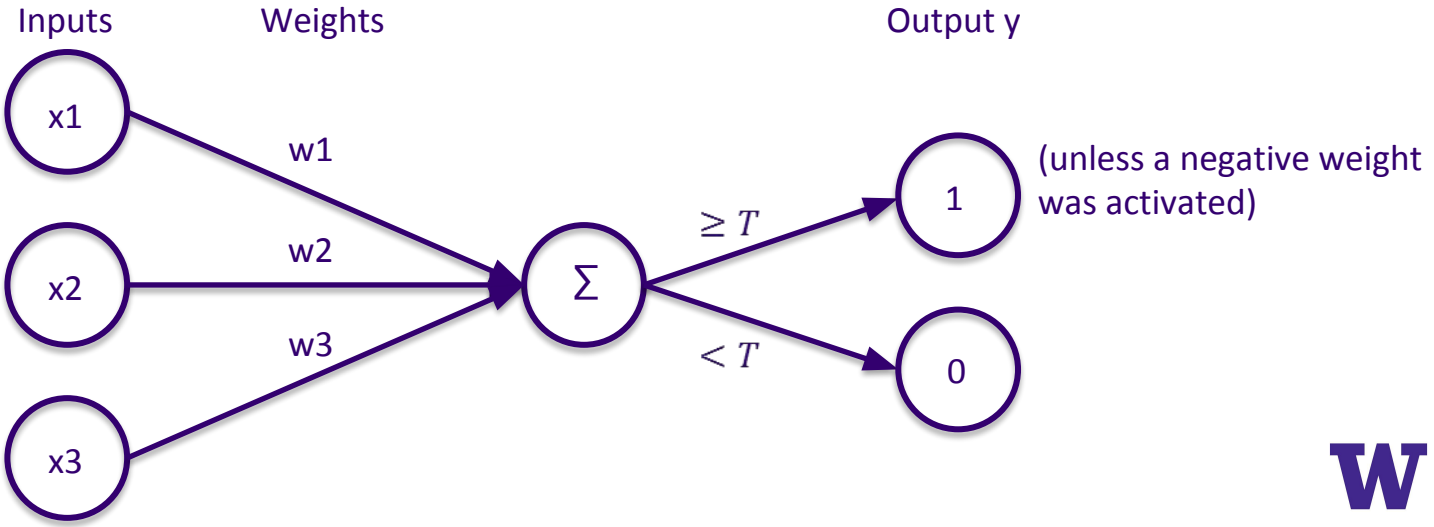
- 1943 Paper: "A logical calculus of the ideas immanent in nervous activity"
  - > Explained how simple neural building blocks could compute functions
- Modern neural networks are still hugely influenced by ideas in the original MCP model

> MCP model

- Neurons at a particular time either send a signal (value 1) or don't send a signal (value 0)
- For a neuron receiving a signal, each synapse has a floating point "weight", which is multiplied by the incoming signal.
- Weights may be excitatory (positive) or inhibitory (negative)

McCollough-Pitts (MCP) Model

- Weighted inputs to a neuron are added together:
  - Input signal $z = \sum_{i=1}^{n} w_i x_i = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n$
- A threshold *T* is used to determine the final output *y:*
  - $y = 1 \;\; if \; z \geq T \; and \; 0 \;\; if \; z < T$
- If there is any negative component in the sum, also set $y = 0$.



McCollough-Pitts (MCP) Model Example

> Consider an MCP neuron with 3 input synapses, with weights [0.5, 0.3, 0.8], and threshold 0.9.
  - If the inputs are [1, 0, 1], what is the output?
  - If the inputs are [1, 1, 0], what is the output?

> Consider an MCP neuron with 3 input synapses, with weights [0.5, -0.3, 0.8], and threshold 0.9.
  - If the inputs are [1, 0, 1], what is the output?
  - If the inputs are [1, 1, 0], what is the output?

Introduction and History of Neural Networks

The Perceptron

- Frank Rosenblatt (1958) described a model similar to McCollough-Pitts that could learn the weights (w1, w2,…) *itself*, based on training examples.
- This was a significant advance in machine learning and artificial intelligence (AI). The field of AI had just been named at this time, in 1956 at the Dartmouth Conference.

Perceptron Model

≫Almost like MCP but with threshold T=0:

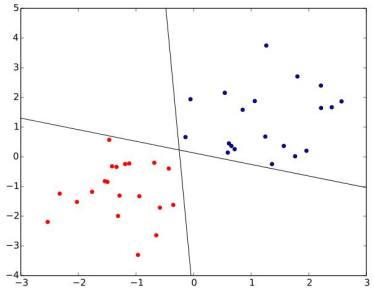
$$z = \sum_{i=1}^{n} w_i x_i = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n$$

>Output $y = 1 \quad if\ z > 0;$
$0 \quad if\ z \leq 0$

>No special rule about negative weights.

Perceptron

- Perceptron Training Algorithm
  1. Initialize the weights to 0 or a small random value
  2. For each training example $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]$, in a training dataset with target value t:
     a) Calculate the output y
     b) Update each weight: $w_i \mathrel{+}= (t - y)x_i$ for all i
  3. Repeat step 2 until we reach a desired error rate or until we hit a specified max # of iterations

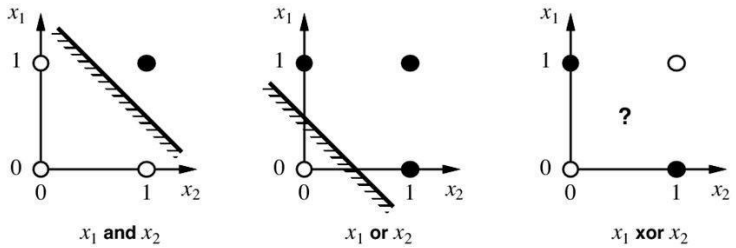


- This is **guaranteed** to converge if the input examples are *linearly separable*

Perceptron: Limitations

x1	x2	Y
0	0	0
0	1	1
1	0	1
1	1	0

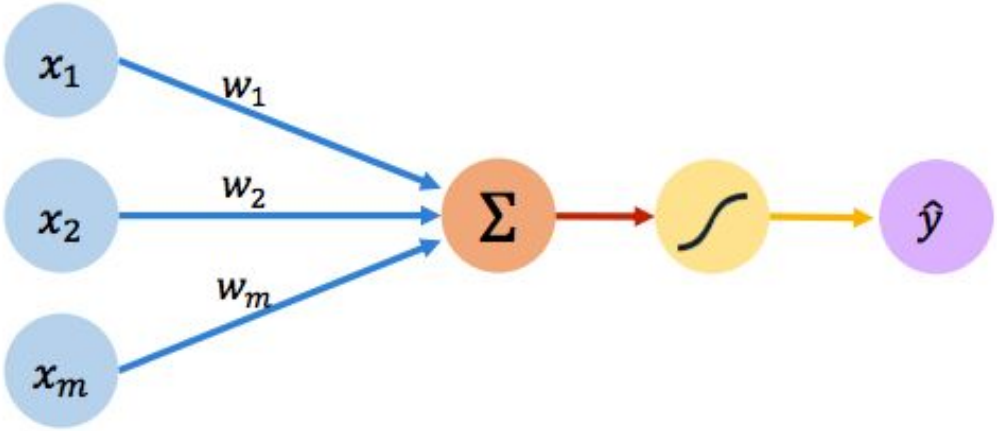
- Minsky and Papert 1969: *Perceptrons: an Introduction to Computational Geometry.*
- Showed that a simple perceptron is unable to represent the XOR function.
- Perceptrons require input example categories to be linearly separable.
  - OR and AND are linearly separable
  - XOR isn't.



Introduction and History of Neural Networks

- What was missing in the perceptron?
- Two things:
  - Multiple layers of units
  - Simpler Nonlinearity
- Note that multiple layers of *linear* neurons (i.e., without the threshold piece) would still just behave like a single layer. Nonlinearity is required to get interesting behavior.

The Perceptron: Forward Propagation



Inputs Weights Sum Non-Linearity Output

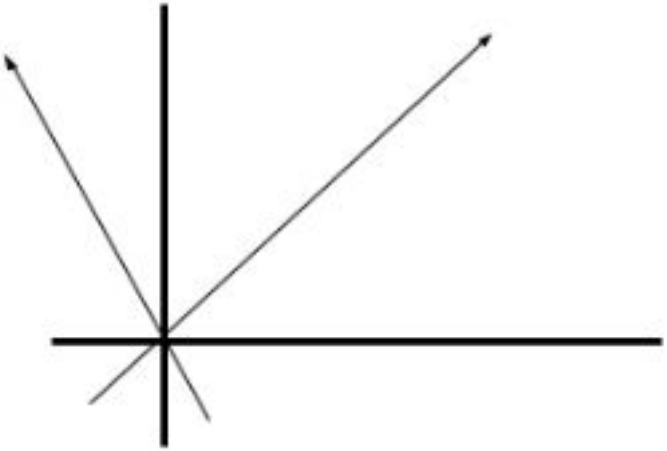
Output

Linear combination of inputs

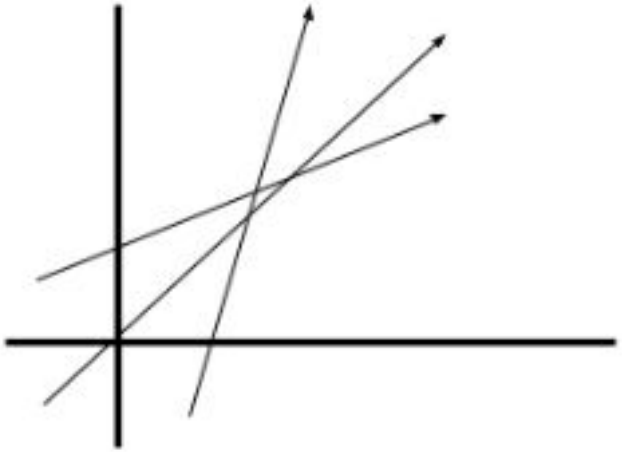
$$\hat{y} = g\left(\sum_{i=1}^{m} x_i \, w_i\right)$$

Non-linear activation function

The Perceptron: Bias Term

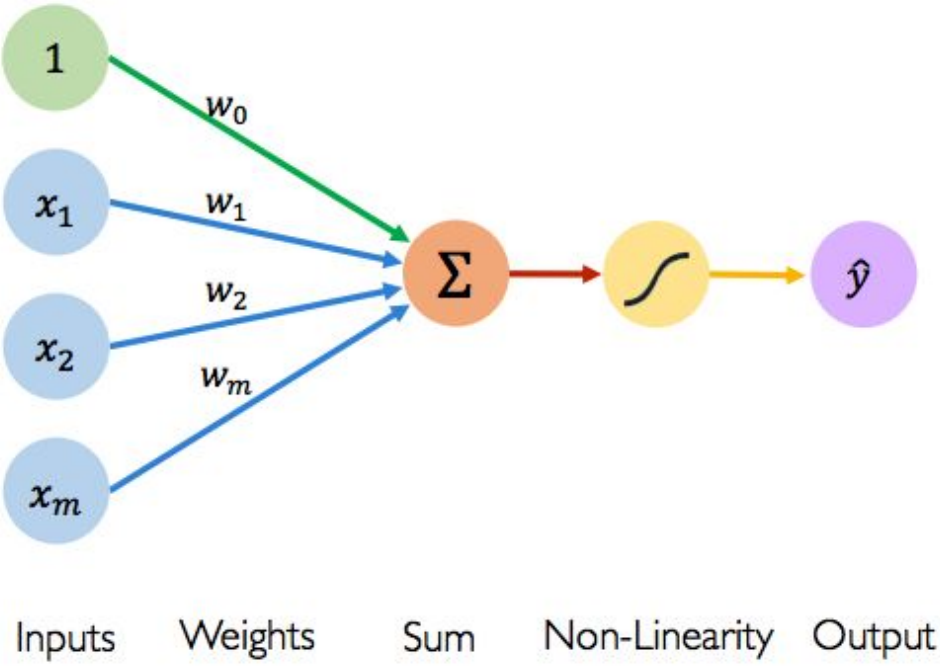


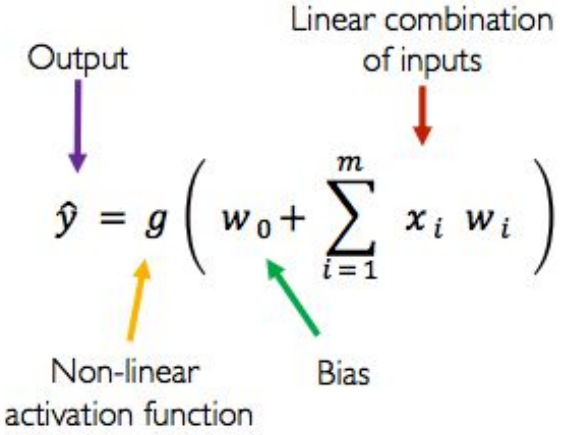
Y = mx



Y = mx + c

The Perceptron: Forward Propagation

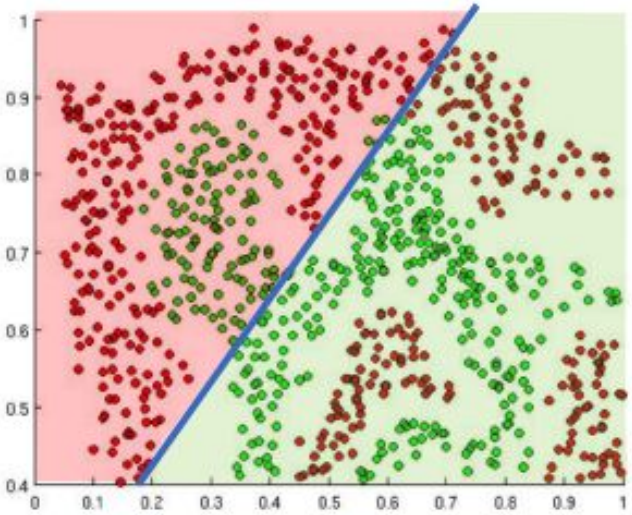




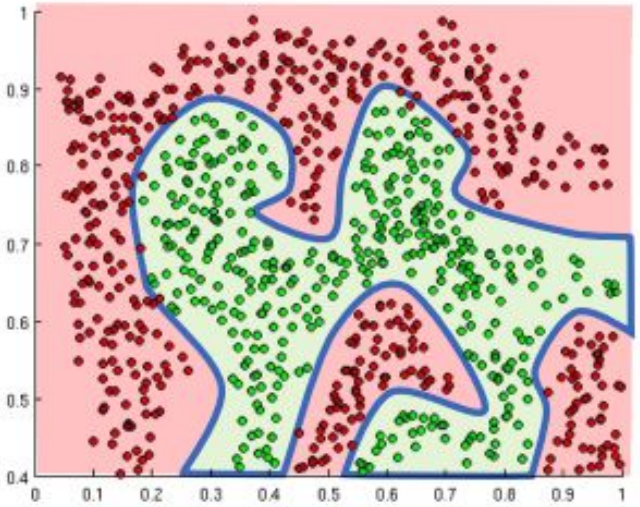
Foster School of Business

Importance of Activation Functions

The purpose of activation functions is to introduce non-linearities into the network



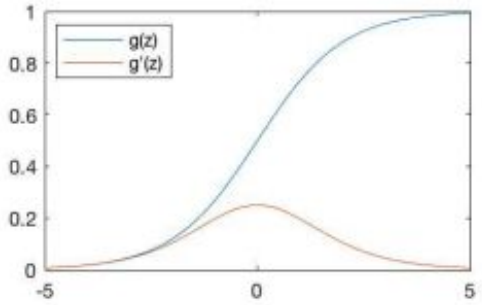
Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

Common Activation Functions

Sigmoid Function

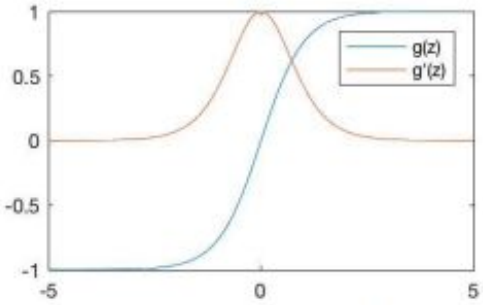


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

tf.nn.sigmoid(z)

Hyperbolic Tangent

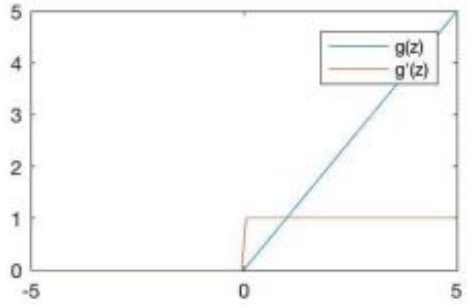


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

tf.nn.tanh(z)

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

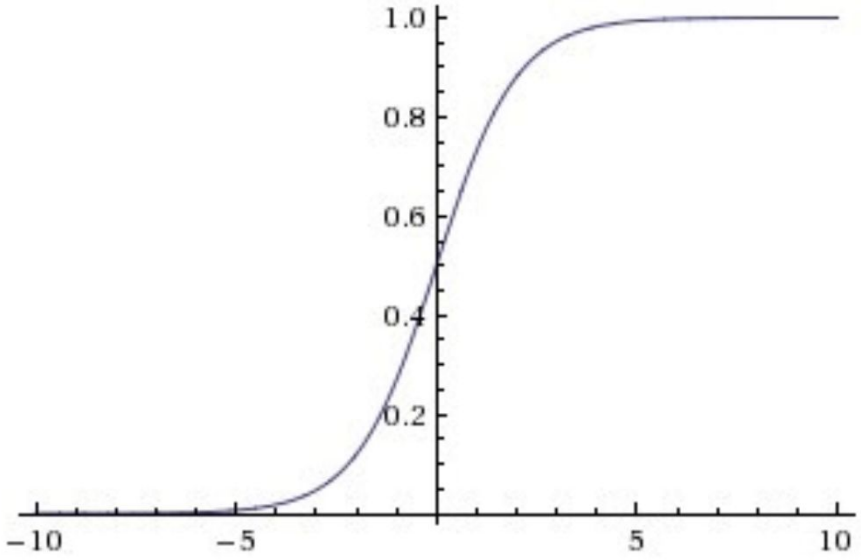
tf.nn.relu(z)

Sigmoid Function

Sigmoid

$$\sigma(x)=1/(1+\exp(-x))$$

- Range [0,1]
- No longer recommended for hidden layer activations
- When used as output activation can be interpreted as a probability

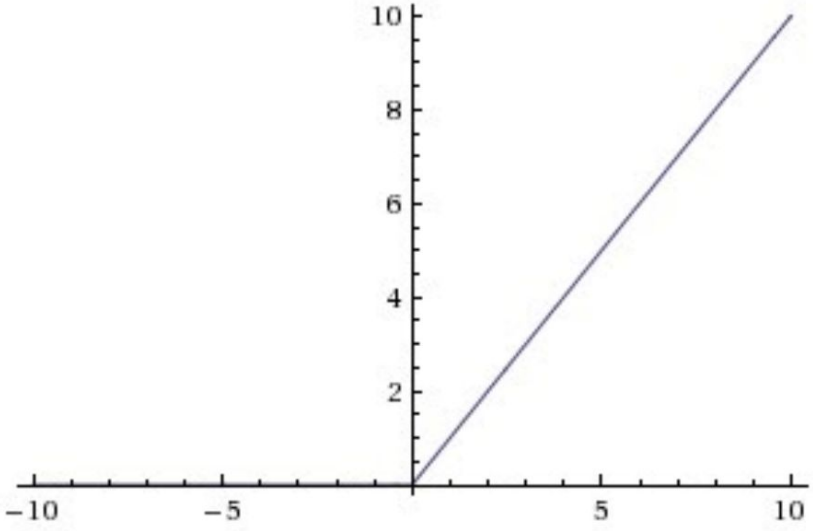


Rectified Linear Unit (ReLU) Function

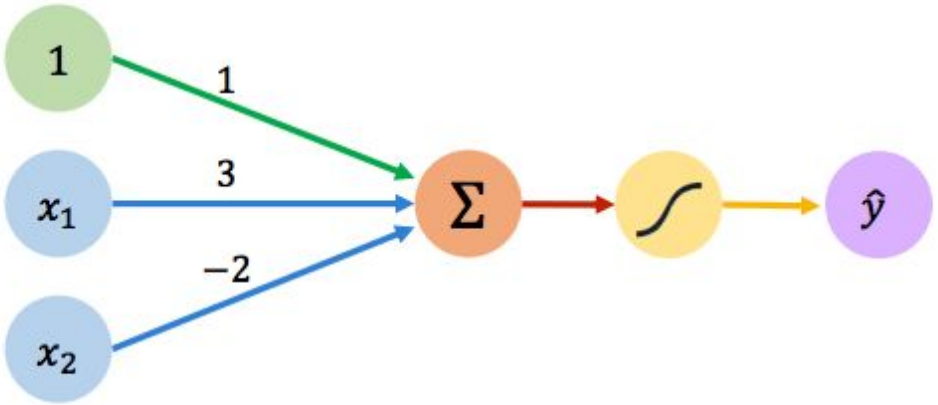
Rectified Linear Unit (ReLU)

f(x) = max(0,x)

- Range: [0,∞)
- Default best choice
- Gradient is 1 if out is positive, else 0
- Effectively a gradient switch



Perceptron Example



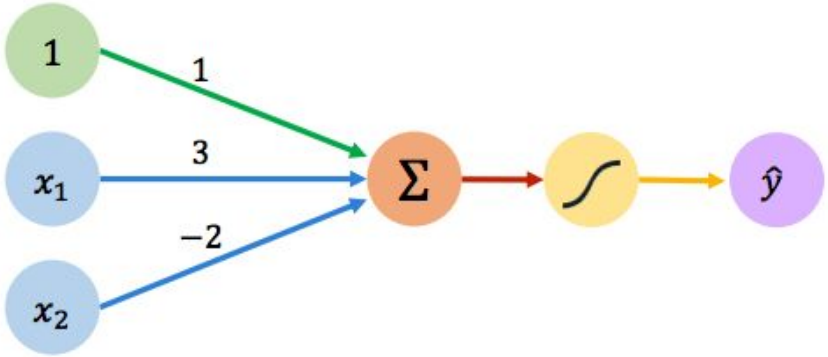
We have: $w_0 = 1$ and $\boldsymbol{W} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g(w_0 + \boldsymbol{X}^T \boldsymbol{W})$$

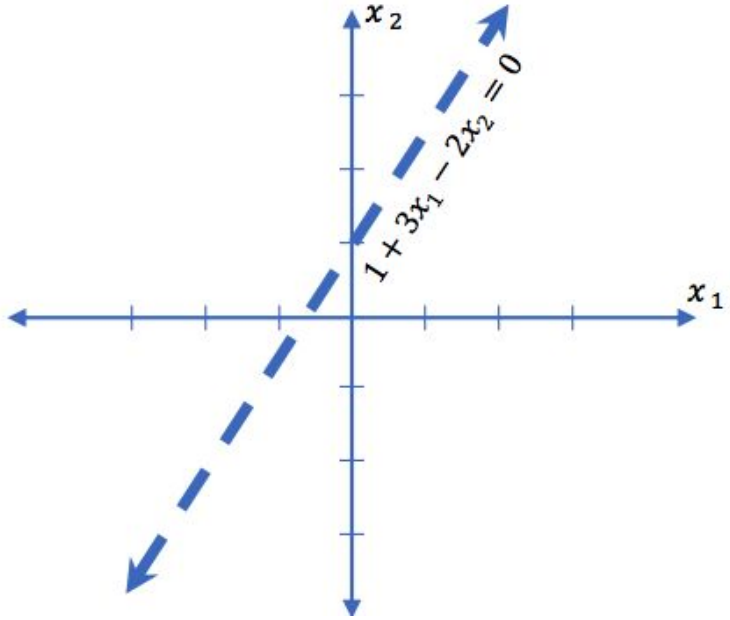
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$

$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

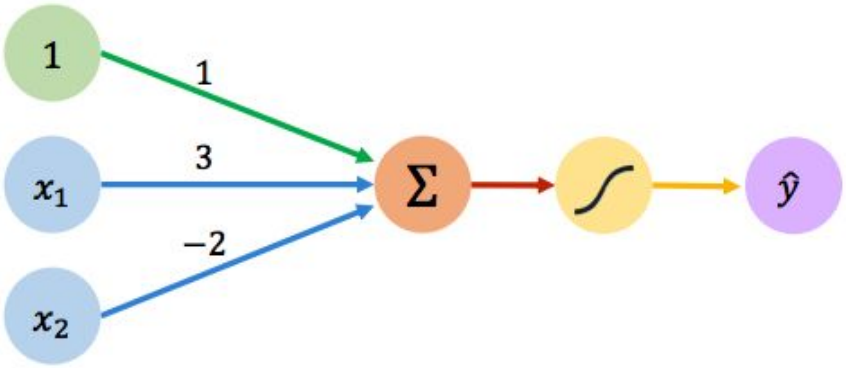
Perceptron Example



$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



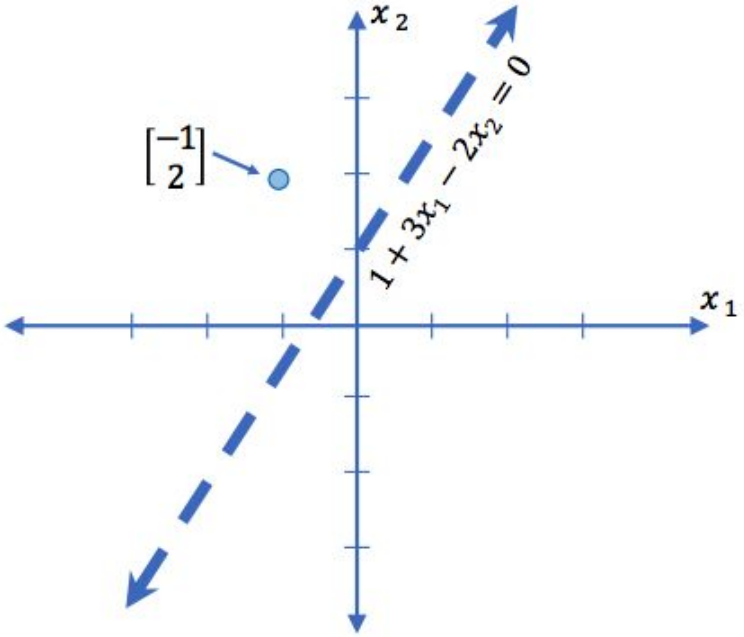
Perceptron Example



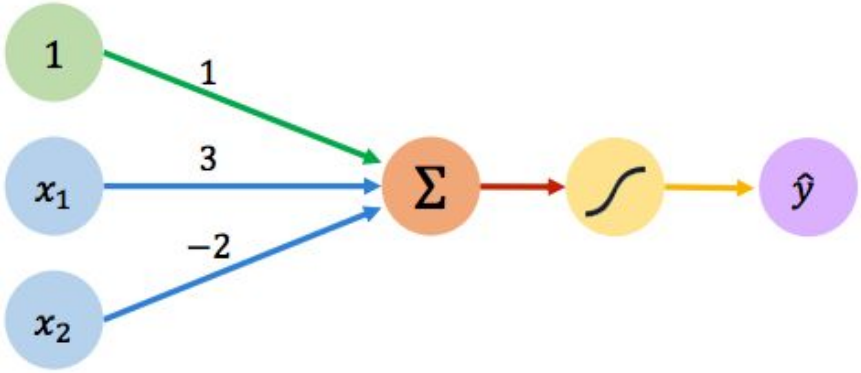
Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

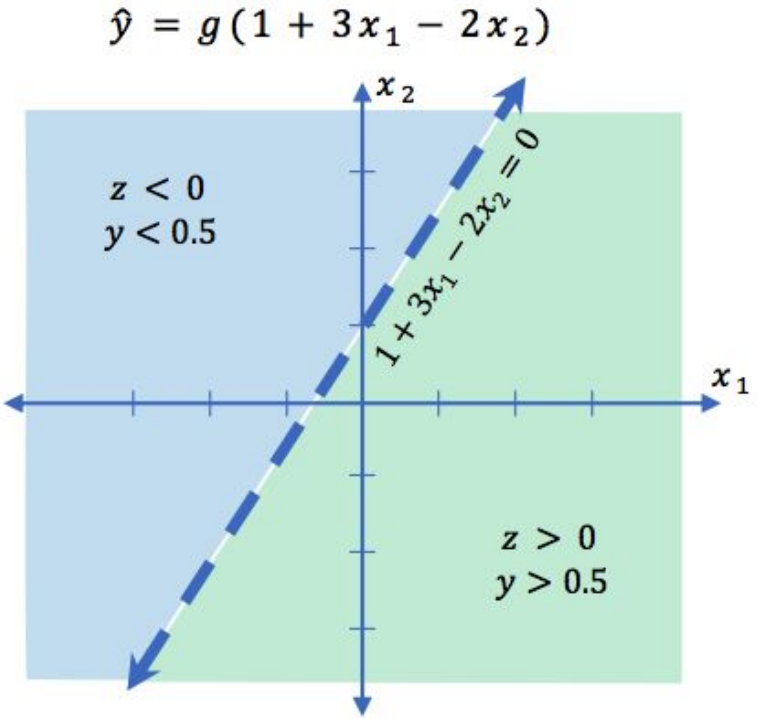
$$\hat{y} = g\left(1 + (3 * -1) - (2 * 2)\right)$$
$$= g(-6) \approx 0.002$$

$$\hat{y} = g(1 + 3x_1 - 2x_2)$$

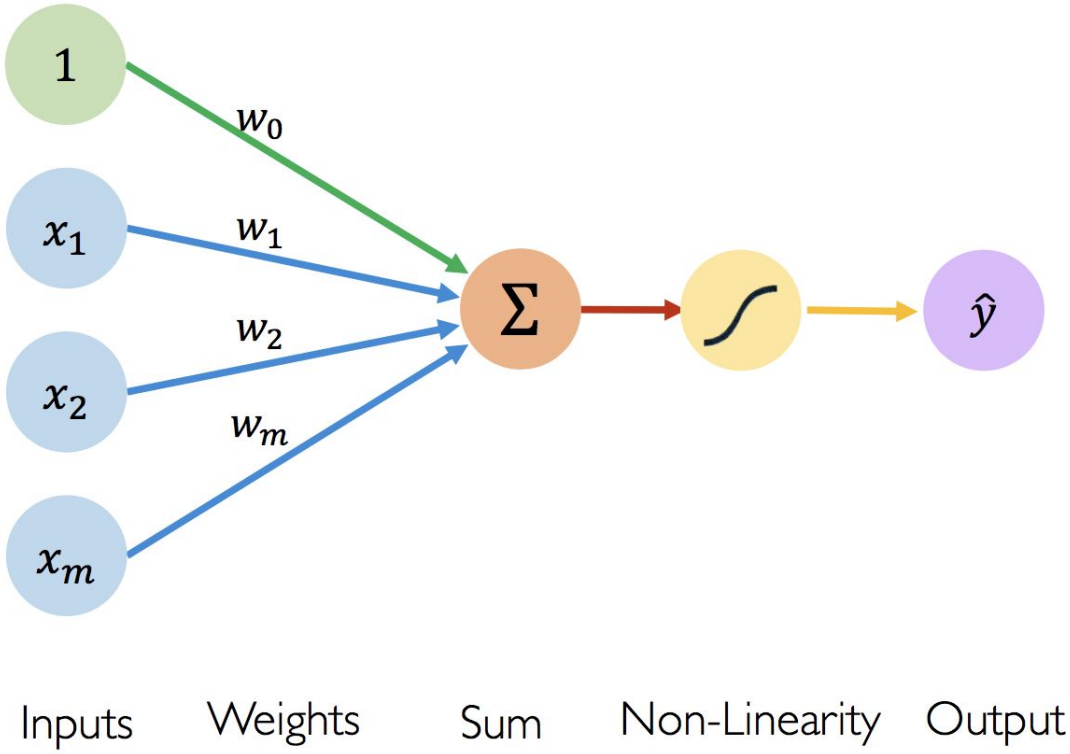


Perceptron Example

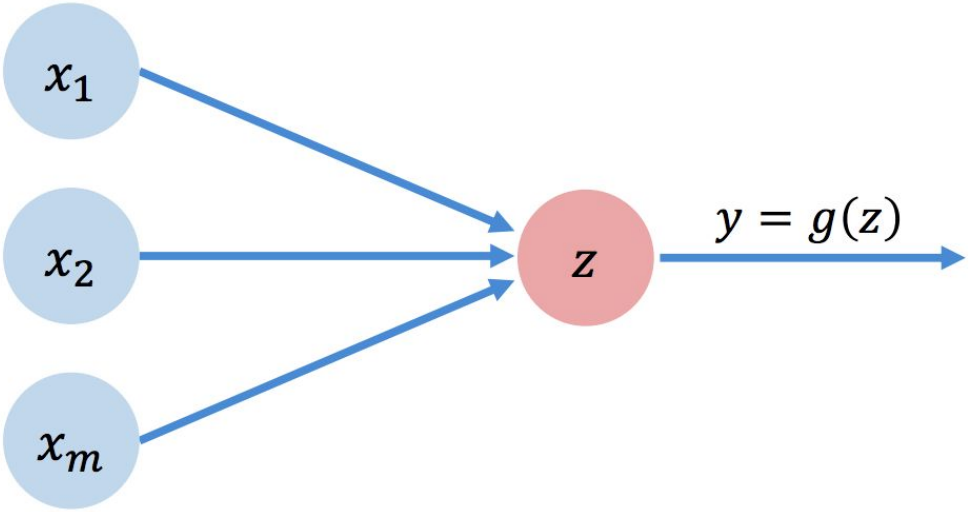




Build Neural Networks with Perceptrons

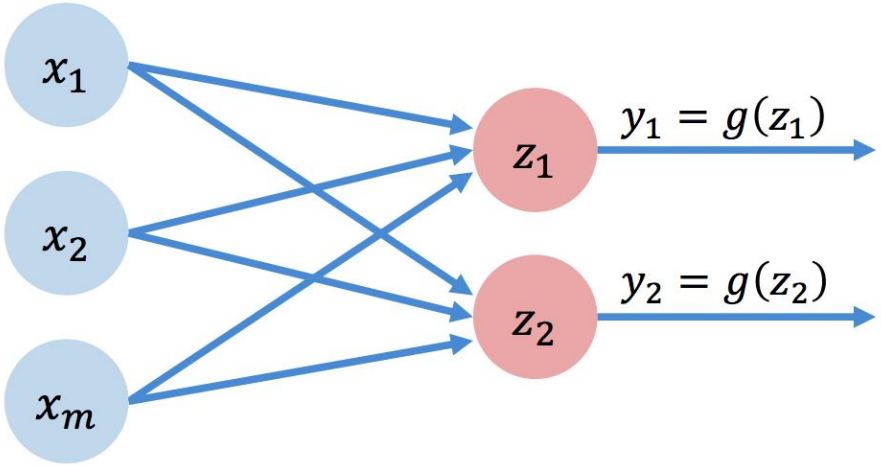


Build Neural Networks with Perceptrons



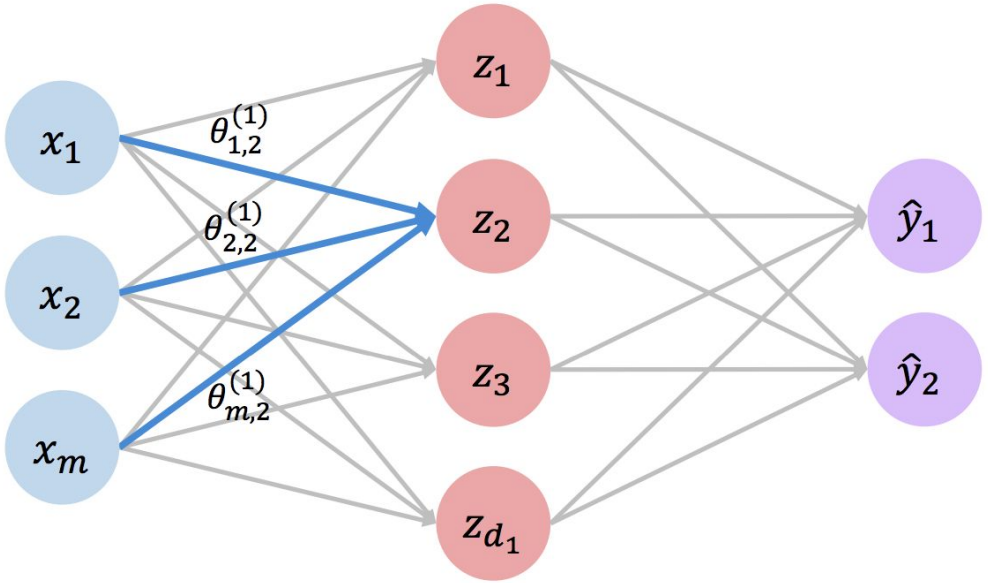
$$z = w_0 + \sum_{j=1}^{m} x_j w_j$$

Build Neural Networks with Perceptrons



$$z_i = w_{0,i} + \sum_{j=1}^{m} x_j \, w_{j,i}$$

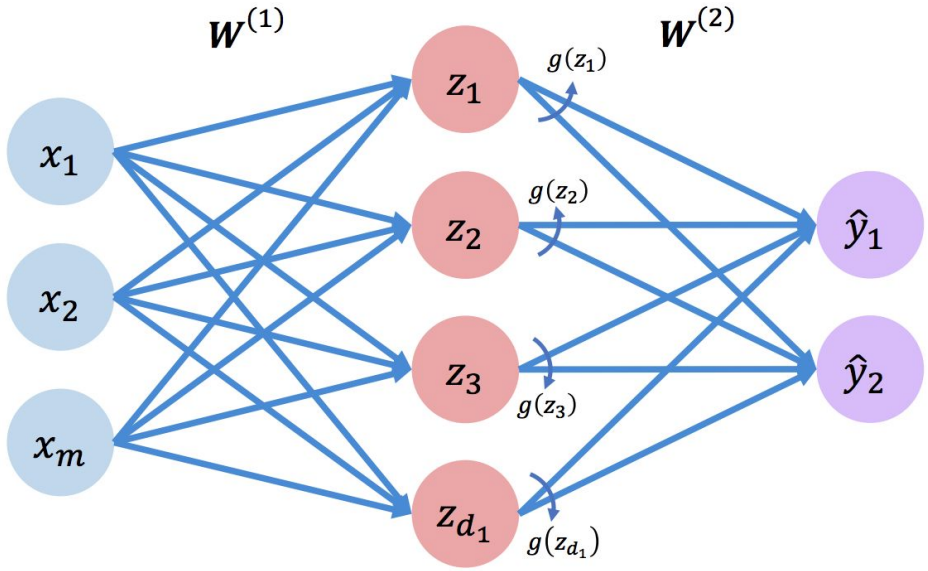
Build Neural Networks with Perceptrons



$$z_2 = w_{0,2}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,2}^{(1)}$$

$$= w_{0,2}^{(1)} + x_1 \, w_{1,2}^{(1)} + x_2 \, w_{2,2}^{(1)} + x_m \, w_{m,2}^{(1)}$$

Build Neural Networks with Perceptrons

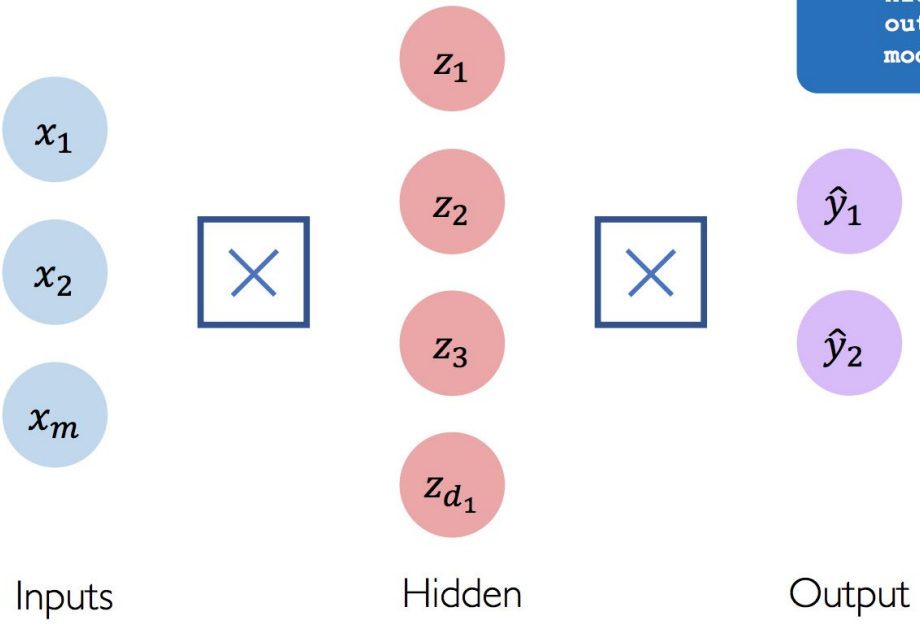


$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j \, w_{j,i}^{(1)} \qquad \hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j \, w_{j,i}^{(2)}\right)$$

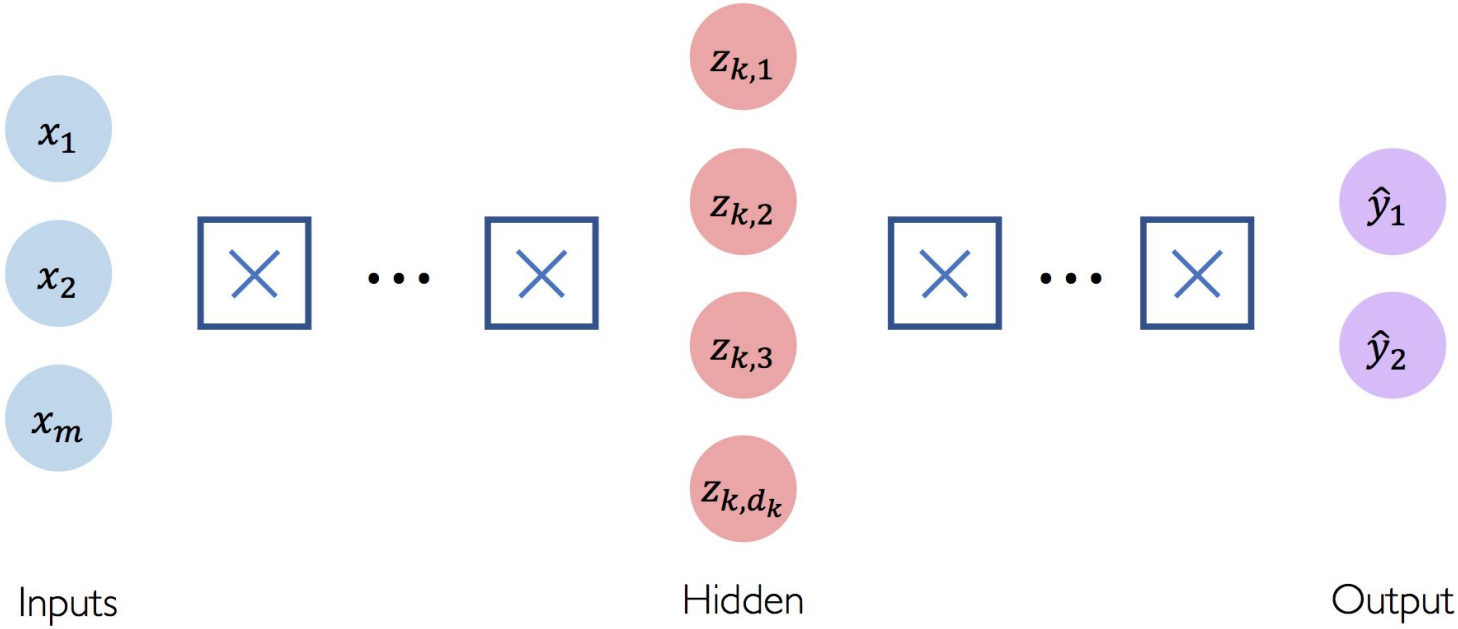
Build Neural Networks with Perceptrons

```
from tf.keras.layers import *

inputs = Inputs(m)
hidden = Dense(d1)(inputs)
outputs = Dense(2)(hidden)
model = Model(inputs, outputs)
```

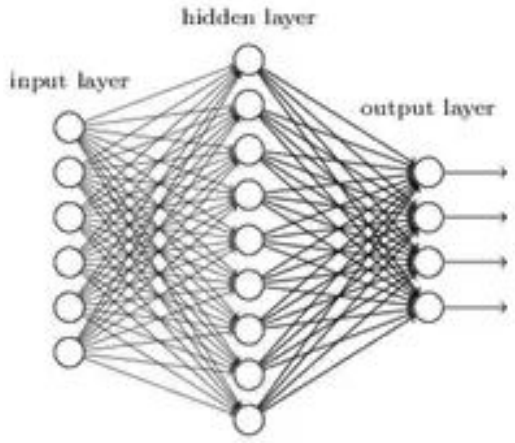


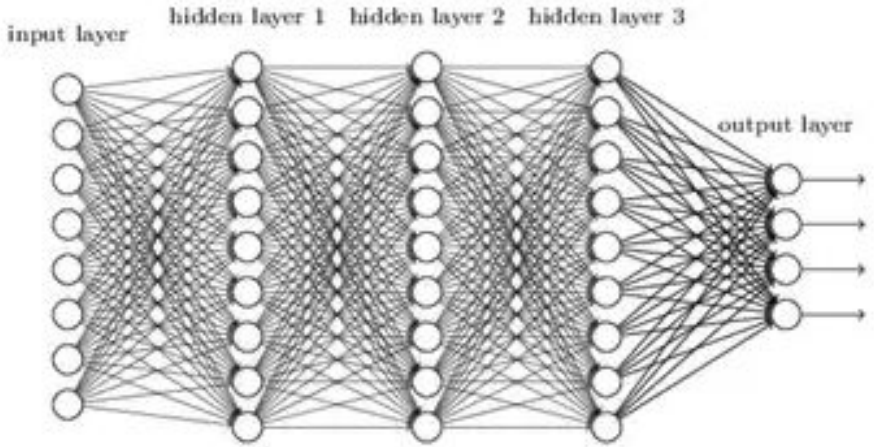
Build Neural Networks with Perceptrons



$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g(z_{k-1,j})\, w_{j,i}^{(k)}$$

Shallow vs. Deep Networks

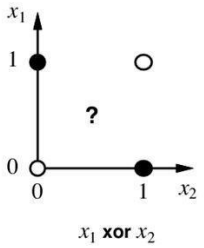




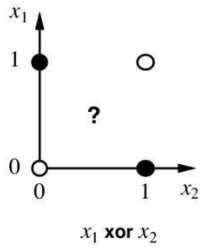
Advantages of Deep Networks:

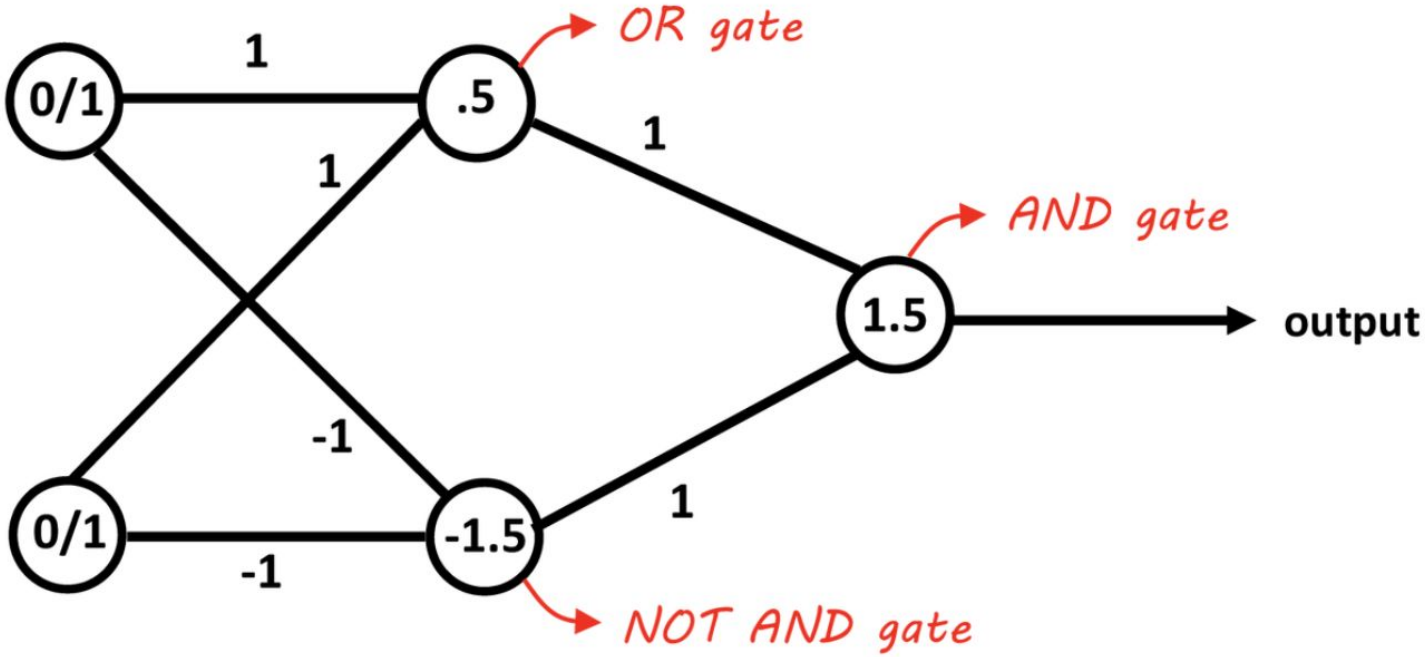
- Deep representation allows for a hierarchy of representations.
- Very wide and shallow networks are very good at memorization while deep networks are better at generalization.
- Often outperform shallow ones with the same computational power.

Design a 1 hidden layer NN for XOR

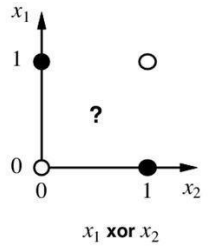


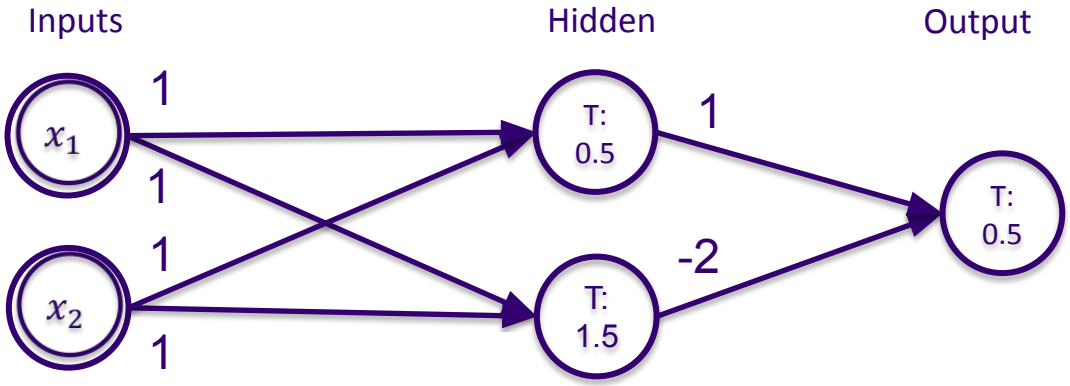
Design a 1 hidden layer NN for XOR





Design a 1 hidden layer NN for XOR





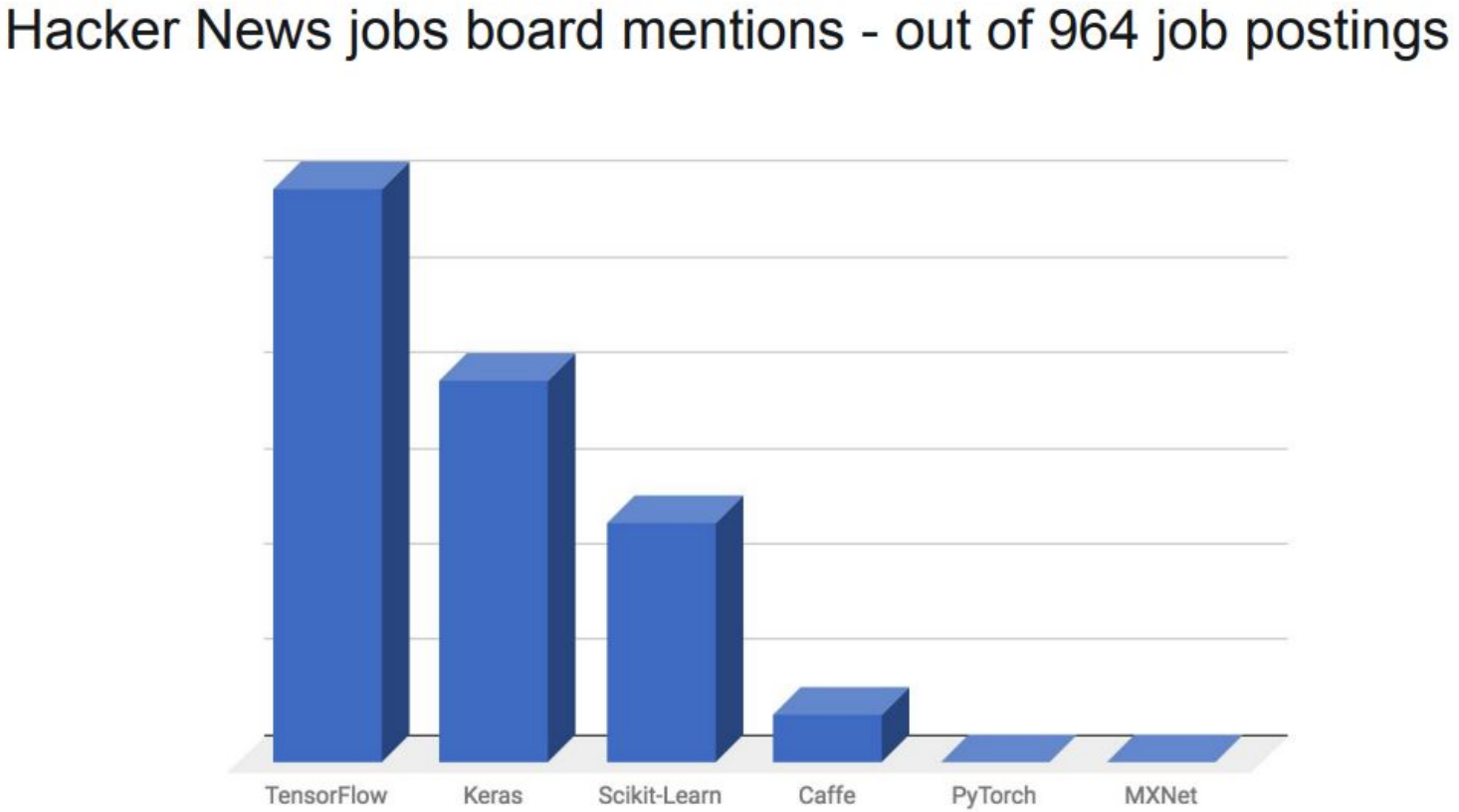
Example trained multi-layer perceptron, with weights shown on connections.

"T:" specifies a neuron with modified threshold: e.g., 0.5 instead of 0

I.e., output $y = 1 \quad if\ z > T;$
$0 \quad if\ z \leq T$

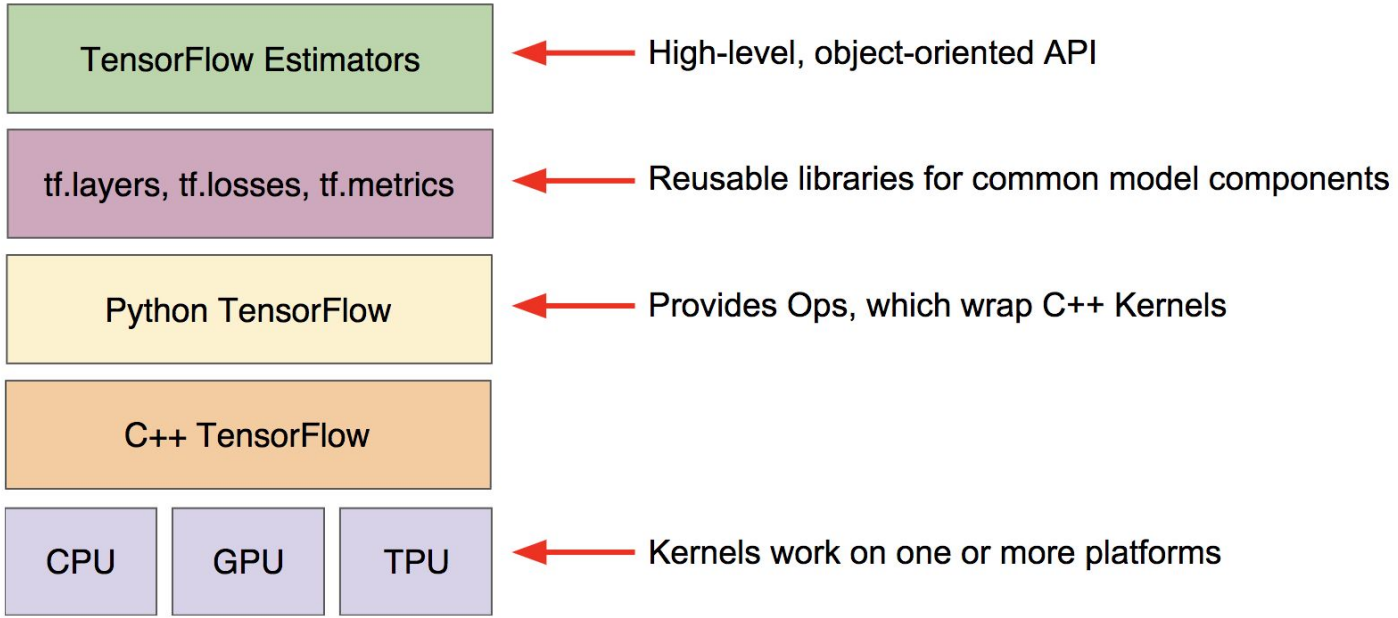
Note that multiple layers of *linear* neurons (i.e., without the threshold piece) would still just behave like a single layer. Nonlinearity is required to get interesting behavior.

Industry Job Demand on DL



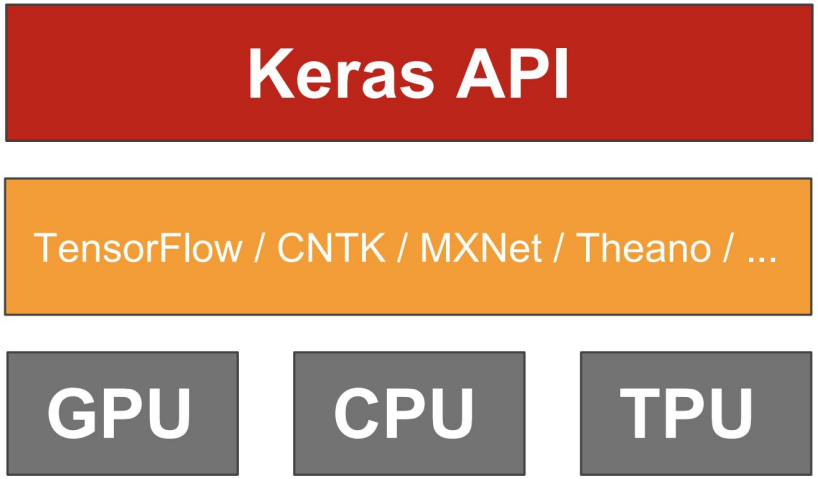
What is Tensorflow?

Tensorflow is a computational framework for building machine learning models.



What is Keras?

- Keras is the official high-level API of TensorFlow
  - Part of core TensorFlow since v1.4
  - Full Keras API
  - Better optimized for TF
- Multi-backend, multi-platform
- Easy to learn and easy to use



Three Keras API styles

- The Sequential Model
  - Dead simple
  - Only for single-input, single-output, sequential layer stacks
  - Good for 70+% of use cases
- The functional API
  - Like playing with Lego bricks
  - Multi-input, multi-output, arbitrary static graph topologies
  - Good for 95% of use cases
- Model subclassing
  - Maximum flexibility
  - Larger potential error surface

The Sequential API

```
import keras
from keras import layers

model = keras.Sequential()
model.add(layers.Dense(20, activation='relu', input_shape=(10,)))
model.add(layers.Dense(20, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.fit(x, y, epochs=10, batch_size=32)
```

The functional API

```
import keras
from keras import layers

inputs = keras.Input(shape=(10,))
x = layers.Dense(20, activation='relu')(x)
x = layers.Dense(20, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)

model = keras.Model(inputs, outputs)
model.fit(x, y, epochs=10, batch_size=32)
```

Model subclassing

```
import keras
from keras import layers

class MyModel(keras.Model):

    def __init__(self):
        super(MyModel, self).__init__()
        self.dense1 = layers.Dense(20, activation='relu')
        self.dense2 = layers.Dense(20, activation='relu')
        self.dense3 = layers.Dense(10, activation='softmax')

    def call(self, inputs):
        x = self.dense1(x)
        x = self.dense2(x)
        return self.dense3(x)

model = MyModel()
model.fit(x, y, epochs=10, batch_size=32)
```

What is Colaboratory?

- Colaboratory is a FREE research tool from Google for machine learning education and research.
- It's a Jupyter notebook environment that requires no setup to use (access through the browsers).
- All Colaboratory notebooks are stored in Google Drive.

https://colab.research.google.com

https://www.youtube.com/watch?v=inN8seMm7UI

Lab